Exploring Application-Level Semantics for Data Compression

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

Abstract—Natural phenomena show that many creatures form large social groups and move in regular patterns. However, previous works focus on finding the movement patterns of each single object or all objects. In this paper, we first propose an efficient distributed mining algorithm to jointly identify a group of moving objects and discover their movement patterns in wireless sensor networks. Afterward, we propose a compression algorithm, called 2P2D, which exploits the obtained group movement patterns to reduce the amount of delivered data. The compression algorithm includes a sequence merge and an entropy reduction phases. In the sequence merge phase, we propose a Merge algorithm to merge and compress the location data of a group of moving objects. In the entropy reduction phase, we formulate a Hit Item Replacement (HIR) problem and propose a Replace algorithm that obtains the optimal solution. Moreover, we devise three replacement rules and derive the maximum compression ratio. The experimental results show that the proposed compression algorithm leverages the group movement patterns to reduce the amount of delivered data effectively and efficiently.

Index Terms—Data compression, distributed clustering, object tracking.

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 a large amount of location data, and thus, lead to transmission and storage challenges, especially in resource-constrained environments like WSNs. To reduce the data volume, various algorithms have been proposed for data compression and data aggregation [1], [2], [3], [4], [5], [6]. However, the above works do not address application-level semantics, such as the group relationships and movement patterns, in the location data.

In object tracking applications, many natural phenomena show that objects often exhibit some degree of regularity in their movements. For example, the famous annual wildebeest migration demonstrates that the movements of creatures are temporally and spatially correlated. Biologists also have found that many creatures, such as elephants, zebra, whales, and birds, form large social groups when migrating to find food, or for breeding or wintering. These characteristics indicate that the trajectory data of multiple objects may be correlated for biological applications. Moreover, some research domains, such as the study of animals' social behavior and wildlife migration [7], [8], are more concerned with the movement patterns of groups of animals, not individuals; hence, tracking each object is unnecessary in this case. This raises a new challenge of finding moving animals belonging to the same group and identifying their aggregated group movement patterns. Therefore, under the assumption that objects with similar movement patterns are regarded as a group, we define the moving object clustering problem as given the movement trajectories of objects, partitioning the objects into non-overlapped groups such that the number of groups is minimized. Then, group movement pattern discovery is to find the most representative movement patterns regarding each group of objects, which are further utilized to compress location data.

Discovering the group movement patterns is more difficult than finding the patterns of a single object or all objects, because we need to jointly identify a group of objects and discover their aggregated group movement patterns. The constrained resource of WSNs should also be considered in approaching the moving object clustering problem. However, few of existing approaches consider these issues simultaneously. On the one hand, the temporal-and-spatial correlations in the movements of moving objects are modeled as sequential patterns in data mining to discover the frequent movement patterns [9], [10], [11], [12]. However, sequential patterns 1) consider the characteristics of *all* objects, 2) lack information about a frequent pattern's significance regarding individual trajectories, and 3) carry no time information between consecutive items, which make them unsuitable for location prediction and similarity comparison. On the other hand, previous works, such as

- *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, and the Department of Electrical Engineering (EE), Department of Computer Science and Information Engineer (CSIE), and 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 22 Sept. 2008; revised 27 Feb. 2009; accepted 29 July 2009; published online 4 Feb. 2010.
Recommended for acceptance by M. Ester.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2008-09-0497.
Digital Object Identifier no. 10.1109/TKDE.2010.30.

[13], [14], [15], measure the similarity among these entire trajectory sequences to group moving objects. Since objects may be close together in some types of terrain, such as gorges, and widely distributed in less rugged areas, their group relationships are distinct in some areas and vague in others. Thus, approaches that perform clustering among entire trajectories may not be able to identify the local group relationships. In addition, most of the above works are centralized algorithms [9], [10], [11], [12], [13], [14], [15], which need to collect all data to a server before processing. Thus, unnecessary and redundant data may be delivered, leading to much more power consumption because data transmission needs more power than data processing in WSNs [5]. In [16], we have proposed a clustering algorithm to find the group relationships for query and data aggregation efficiency. The differences of [16] and this work are as follows: First, since the clustering algorithm itself is a centralized algorithm, in this work, we further consider systematically combining multiple local clustering results into a consensus to improve the clustering quality and for use in the update-based tracking network. Second, when a delay is tolerant in the tracking application, a new data management approach is required to offer transmission efficiency, which also motivates this study. We thus define the problem of compressing the location data of a group of moving objects as the group data compression problem.

Therefore, in this paper, we first introduce our distributed mining algorithm to approach the moving object clustering problem and discover group movement patterns. Then, based on the discovered group movement patterns, we propose a novel compression algorithm to tackle the group data compression problem. Our distributed mining algorithm comprises a Group Movement Pattern Mining (GMPMine) and a Cluster Ensembling (CE) algorithms. It avoids transmitting unnecessary and redundant data by transmitting only the local grouping results to a base station (the sink), instead of all of the moving objects' location data. Specifically, the GMPMine algorithm discovers the local group movement patterns by using a novel similarity measure, while the CE algorithm combines the local grouping results to remove inconsistency and improve the grouping quality by using the information theory.

Different from previous compression techniques that remove redundancy of data according to the regularity within the data, we devise a novel two-phase and 2D algorithm, called 2P2D, which utilizes the discovered group movement patterns shared by the transmitting node and the receiving node to compress data. In addition to remove redundancy of data according to the correlations within the data of each single object, the 2P2D algorithm further leverages the correlations of multiple objects and their movement patterns to enhance the compressibility. Specifically, the 2P2D algorithm comprises a sequence merge and an entropy reduction phases. In the sequence merge phase, we propose a Merge algorithm to merge and compress the location data of a group of objects. In the entropy reduction phase, we formulate a Hit Item Replacement (HIR) problem to minimize the entropy of the merged data and propose a Replace algorithm to obtain the optimal solution. The Replace algorithm finds the optimal solution of the HIR problem based on Shannon's theorem [17] and guarantees the reduction of entropy, which is conventionally viewed as an optimization bound of compression performance. As a result, our approach reduces the amount of delivered data and, by extension, the energy consumption in WSNs.

Our contributions are threefold:

- Different from previous works, we formulate a moving object clustering problem that jointly identifies a group of objects and discovers their movement patterns. The application-level semantics are useful for various applications, such as data storage and transmission, task scheduling, and network construction.
- To approach the moving object clustering problem, we propose an efficient distributed mining algorithm to minimize the number of groups such that members in each of the discovered groups are highly related by their movement patterns.
- We propose a novel compression algorithm to compress the location data of a group of moving objects with or without loss of information. We formulate the HIR problem to minimize the entropy of location data and explore the Shannon's theorem to solve the HIR problem. We also prove that the proposed compression algorithm obtains the optimal solution of the HIR problem efficiently.

The remainder of the paper is organized as follows: In Section 2, we review related works. In Section 3, we provide an overview of the network, location, and movement models and formulate our problem. In Section 4, we describe the distributed mining algorithm. In Section 5, we formulate the compression problems and propose our compression algorithm. Section 6 details our experimental results. Finally, we summarize our conclusions in Section 7.

2 Related Work

2.1 Movement Pattern Mining

Agrawal and Srikant [18] first defined the sequential pattern mining problem and proposed an Apriori-like algorithm to find the frequent sequential patterns. Han et al. consider the pattern projection method in mining sequential patterns and proposed FreeSpan [19], which is an FP-growth-based algorithm. Yang and Hu [9] developed a new match measure for imprecise trajectory data and proposed TrajPattern to mine sequential patterns. Many variations derived from sequential patterns are used in various applications, e.g., Chen et al. [20] discover path traversal patterns in a Web environment, while Peng and Chen [21] mine user moving patterns incrementally in a mobile computing system. However, sequential patterns and its variations like [20], [21] do not provide sufficient information for location prediction or clustering. First, they carry no time information between consecutive items, so they cannot provide accurate information for location prediction when time is concerned. Second, they consider the characteristics of *all* objects, which make the meaningful movement characteristics of individual objects or a group of moving objects inconspicuous and ignored. Third, because

a sequential pattern lacks information about its significance regarding to each individual trajectory, they are not fully representative to individual trajectories. To discover significant patterns for location prediction, Morzy mines frequent trajectories whose consecutive items are also adjacent in the original trajectory data [10], [22]. Meanwhile, Giannotti et al. [11] extract T-patterns from spatiotemporal data sets to provide concise descriptions of frequent movements, and Tseng and Lin [12] proposed the TMP-Mine algorithm for discovering the temporal movement patterns. However, the above Apriori-like or FP-growth-based algorithms still focus on discovering frequent patterns of *all* objects and may suffer from computing efficiency or memory problems, which make them unsuitable for use in resource-constrained environments.

2.2 Clustering

Recently, clustering based on objects' movement behavior has attracted more attention. Wang et al. [14] transform the location sequences into a transaction-like data on users and based on which to obtain a valid group, but the proposed AGP and VG growth are still Apriori-like or FP-growth-based algorithms that suffer from high computing cost and memory demand. Nanni and Pedreschi [15] proposed a density-based clustering algorithm, which makes use of an optimal time interval and the average euclidean distance between each point of two trajectories, to approach the trajectory clustering problem. However, the above works discover global group relationships based on the proportion of the time a group of users stay close together to the whole time duration or the average euclidean distance of the entire trajectories. Thus, they may not be able to reveal the local group relationships, which are required for many applications. In addition, though computing the average euclidean distance of two geometric trajectories is simple and useful, the geometric coordinates are expensive and not always available. Approaches, such as EDR, LCSS, and DTW, are widely used to compute the similarity of symbolic trajectory sequences [13], but the above dynamic programming approaches suffer from scalability problem [23]. To provide scalability, approximation or summarization techniques are used to represent original data. Guralnik and Karypis [23] project each sequence into a vector space of sequential patterns and use a vector-based K-means algorithm to cluster objects. However, the importance of a sequential pattern regarding individual sequences can be very different, which is not considered in this work. To cluster sequences, Yang and Wang proposed CLUSEQ [24], which iteratively identifies a sequence to a learned model, yet the generated clusters may overlap which differentiates their problem from ours.

2.3 Data Compression

Data compression can reduce the storage and energy consumption for resource-constrained applications. In [1], distributed source (Slepian-Wolf) coding uses joint entropy to encode two nodes' data individually without sharing any data between them; however, it requires prior knowledge of cross correlations of sources. Other works, such as [2], [4], combine data compression with routing by exploiting cross correlations between sensor nodes to reduce the data size. In [5], a tailed LZW has been proposed to address the memory constraint of a sensor device. Summarization of the original data by regression or linear modeling has been proposed for trajectory data compression [3], [6]. However, the above works do not address application-level semantics in data, such as the correlations of a group of moving objects, which we exploit to enhance the compressibility.

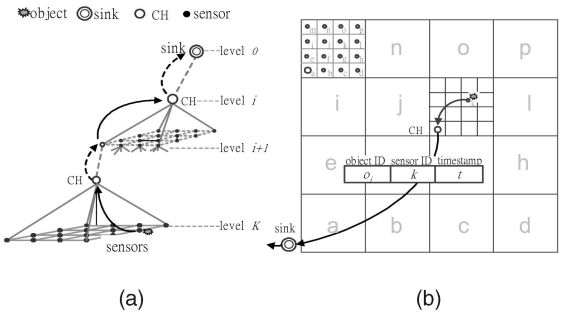


Fig. 1. (a) The hierarchical- and cluster-based network structure and the data flow of an update-based tracking network. (b) A flat view of a two-layer network structure with 16 clusters.

3 Preliminaries

3.1 Network and Location Models

Many researchers believe that a hierarchical architecture provides better coverage and scalability, and also extends the network lifetime of WSNs [25], [26]. In a hierarchical WSN, such as that proposed in [27], the energy, computing, and storage capacity of sensor nodes are heterogeneous. A high-end sophisticated sensor node, such as Intel Stargate [28], is assigned as a cluster head (CH) to perform high complexity tasks; while a resource-constrained sensor node, such as Mica2 mote [29], performs the sensing and low complexity tasks. In this work, we adopt a hierarchical network structure with K layers, as shown in Fig. 1a, where sensor nodes are clustered in each level and collaboratively gather or relay remote information to a base station called a sink. A sensor cluster is a mesh network of $n \times n$ sensor nodes handled by a CH and communicate with each other by using multihop routing [30]. We assume that each node in a sensor cluster has a locally unique ID and denote the sensor IDs by an alphabet Σ. Fig. 1b shows an example of a two-layer tracking network, where each sensor cluster contains 16 nodes identified by $\Sigma = \{a, b, \ldots, p\}$.

In this work, an object is defined as a target, such as an animal or a bird, that is recognizable and trackable by the tracking network. To represent the location of an object, geometric models and symbolic models are widely used [31]. 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 node or a cluster of sensor nodes, defined by the application. Since the accurate geometric location is not easy to obtain and techniques like the Received Signal Strength (RSS) [32] can simply estimate an object's location based on the ID of the sensor node with the strongest signal, we employ a symbolic model and describe the location of an object by using the ID of a nearby sensor node.

Object tracking is defined as a task of detecting a moving object's location and reporting the location data to the sink

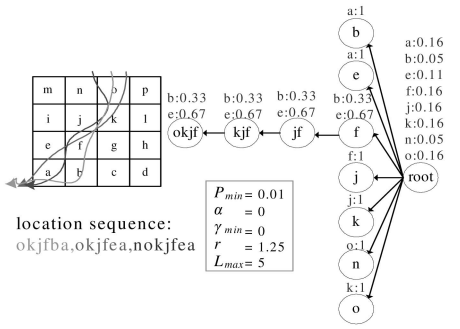


Fig. 2. An example of an object's moving trajectory, the obtained location sequence, and the generated PST T.

periodically at a time interval. Hence, an observation on an object is defined by the obtained location data. We assume that sensor nodes wake up periodically to detect objects. When a sensor node wakes up on its duty cycle and detects an object of interest, it transmits the location data of the object to its CH. Here, the location data include a times tamp, the ID of an object, and its location. We also assume that the targeted applications are delay-tolerant. Thus, instead of forwarding the data upward immediately, the CH compresses the data accumulated for a batch period and sends it to the CH of the upper layer. The process is repeated until the sink receives the location data. Consequently, the trajectory of a moving object is thus modeled as a series of observations and expressed by a location sequence, i.e., a sequence of sensor IDs visited by the object. We denote a location sequence by $S = s_0 s_1 \ldots s_{L-1}$, where each item s_i is a symbol in Σ and L is the sequence length. An example of an object's trajectory and the obtained location sequence is shown in the left of Fig. 2. The tracking network tracks moving objects for a period and generates a location sequence data set, based on which we discover the group relationships of the moving objects.

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

If the movements of an object are regular, the object's next location can be predicted based on its preceding locations. We model the regularity by using the Variable Length Markov Model (VMM). Under the VMM, an object's movement is expressed by a conditional probability distribution over Σ. Let s denote a pattern which is a subsequence of a location sequence S and σ denote a symbol in Σ. The conditional probability $P(\sigma|s)$ is the occurrence probability that σ will follow s in S. Since the length of s is floating, the VMM provides flexibility to adapt to the variable length of movement patterns. Note that when a pattern s occurs more frequently, it carries more information about the movements of the object and is thus more desirable for the purpose of prediction. To find the informative patterns, we first define a pattern as a significant movement pattern if its occurrence probability is above a minimal threshold.

To learn the significant movement patterns, we adapt Probabilistic Suffix Tree (PST) [33] for it has the lowest storage requirement among many VMM implementations [34]. PST's low complexity, i.e., $O(n)$ in both time and space [35], also makes it more attractive especially for streaming or resource-constrained environments [36]. The PST building algorithm[1] learns from a location sequence and generates a PST whose height is limited by a specified parameter L_{max}. Each node of the tree represents a significant movement pattern s whose occurrence probability is above a specified minimal threshold P_{min}. It also carries the conditional empirical probabilities $P(\sigma|s)$ for each σ in Σ that we use in location prediction. Fig. 2 shows an example of a location sequence and the corresponding PST. $node_f$ is one of the children of the root node and represents a significant movement pattern with $"f"$ whose occurrence probability is above 0.01. The conditional empirical probabilities are $P('b'|"f") = 0.33$, $P('e'|"f") = 0.67$, and $P(\sigma|"f") = 0$ for the other σ in Σ.

```
Algorithm: predict_next
Input : T, s
Output : σ
0.  σ = ε
1.  node_cur = root of T
2.  node_next = NULL
3.  repeat
4.      remove tail symbol σ´ from s
5.      node_next = child of node_cur with respect to σ´
6.      if node_next exists then
7.          node_cur = node_next
8.      else
9.          break the loop
10. until (s == ε)
11. σ = the symbol with the highest conditional probability in node_cur
12. return σ
```

Fig. 3. The predict_next algorithm.

PST is frequently used in predicting the occurrence probability of a given sequence, which provides us important information in similarity comparison. 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, the occurrence probability $P^T("nokjfb")$ is computed as follows:

$$\begin{aligned} P^T("nokjfb") &= P^T("n") \times P^T('o'|"n") \times P^T('k'|"no") \\ &\quad \times P^T('j'|"nok") \times P^T('f'|"nokj") \\ &\quad \times P^T('b'|"nokjf") \\ &= P^T("n") \times P^T('o'|"n") \times P^T('k'|"o") \\ &\quad \times P^T('j'|"k") \times P^T('f'|"j") \times P^T('b'|"okjf") \\ &= 0.05 \times 1 \times 1 \times 1 \times 1 \times 0.3 = 0.0165. \end{aligned}$$

PST is also useful and efficient in predicting the next item of a sequence. For a given sequence s and a PST T, our predict_next algorithm as shown in Fig. 3 outputs the most probable next item, denoted by $predict\_next(T, s)$. We demonstrate its efficiency by an example as follow: Given $s = "nokjf"$ and T shown in Fig. 2, the predict_next algorithm traverses the tree to the deepest node $node_{okjf}$ along the path including $node_{root}$, $node_f$, $node_{jf}$, $node_{kjf}$, and $node_{okjf}$. Finally, symbol $'e'$, which has the highest conditional empirical probability in $node_{okjf}$, is returned, i.e., $predict\_next(T, "nokjf") = 'e'$. The algorithm's computational overhead is limited by the height of a PST so that it is suitable for sensor nodes.

1. The PST building algorithm is given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30.

3.3 Problem Description

We formulate the problem of this paper as exploring the group movement patterns to compress the location sequences of a group of moving objects for transmission efficiency. Consider a set of moving objects $O = \{o_1, o_2, \ldots, o_n\}$ and their associated location sequence data set $S = \{S_1, S_2, \ldots, S_n\}$.

Definition 1. *Two objects are similar to each other if their movement patterns are similar. Given the similarity measure function sim_p[2] and a minimal threshold sim_{min}, o_i and o_j are similar if their similarity score $sim_p(o_i, o_j)$ is above sim_{min}, i.e., $sim_p(o_i, o_j) \geq sim_{min}$. The set of objects that are similar to o_i is denoted by $so(o_i) = \{o_j | \forall o_j \in O, sim_p(o_i, o_j) \geq sim_{min}\}$.*

Definition 2. *A set of objects is recognized as a group if they are highly similar to one another. Let g denote a set of objects. g is a group if every object in g is similar to at least a threshold of objects in g, i.e., $\forall o_i \in g$, $\frac{|so(o_i) \cap g|}{|g|} \geq \gamma$, where γ is with default value $\frac{1}{2}$.*[3]

We formally define the moving object clustering problem as follows: Given a set of moving objects O together with their associated location sequence data set S and a minimal similarity threshold sim_{min}, the moving object clustering problem is to partition O into nonoverlapped groups, denoted by $G = \{g_1, g_2, \ldots, g_i\}$, such that the number of groups is minimized, i.e., $|G|$ is minimal. Thereafter, with the discovered group information and the obtained group movement patterns, the group data compression problem is to compress the location sequences of a group of moving objects for transmission efficiency. Specifically, we formulate the group data compression problem as a merge problem and an HIR problem. The merge problem is to combine multiple location sequences to reduce the overall sequence length, while the HIR problem targets to minimize the entropy of a sequence such that the amount of data is reduced with or without loss of information.

4 Mining of Group Movement Patterns

To tackle the moving object clustering problem, we propose a distributed mining algorithm, which comprises the GMPMine and CE algorithms. First, the GMPMine algorithm uses a PST to generate an object's significant movement patterns and computes the similarity of two objects by using sim_p to derive the local grouping results. The merits of sim_p include its accuracy and efficiency: First, sim_p considers the significances of each movement pattern regarding to individual objects so that it achieves better accuracy in similarity comparison. For a PST can be used to predict a pattern's occurrence probability, which is viewed as the significance of the pattern regarding the PST, sim_p thus includes movement patterns' predicted occurrence probabilities to provide fine-grained similarity comparison. Second, sim_p can offer seamless and efficient comparison for the applications with evolving and evolutionary similarity relationships. This is because sim_p can compare the similarity of two data streams only on the changed mature nodes of emission trees [36], instead of all nodes.

To combine multiple local grouping results into a consensus, 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. It trades off the grouping quality against the computation cost by adjusting a partition parameter. In contrast to approaches that perform clustering among the entire trajectories, the distributed algorithm discovers the group relationships in a distributed manner on sensor nodes. As a result, we can discover group movement patterns to compress the location data in the areas where objects have explicit group relationships. Besides, the distributed design provides flexibility to take partial local grouping results into ensembling when the group relationships of moving objects in a specified subregion are interested. Also, it is especially suitable for heterogeneous tracking configurations, which helps reduce the tracking cost, e.g., instead of waking up all sensors at the same frequency, a fine-grained tracking interval is specified for partial terrain in the migration season to reduce the energy consumption. Rather than deploying the sensors in the same density, they are only highly concentrated in areas of interest to reduce deployment costs.

4.1 The Group Movement Pattern Mining (GMPMine) Algorithm

To provide better discrimination accuracy, we propose a new similarity measure sim_p to compare the similarity of two objects. For each of their significant movement patterns, the new similarity measure considers not merely two probability distributions but also two weight factors, i.e., the significance of the pattern regarding to each PST. The similarity score sim_p of o_i and o_j based on their respective PSTs, T_i and T_j, is defined as follows:

$$sim_p(o_i, o_j) = -\log \frac{\sum_{s \in \widetilde{S}} \sqrt{\sum_{\sigma \in \Sigma} (P^{T_i}(s\sigma) - P^{T_j}(s\sigma))^2}}{2L_{max} + \sqrt{2}}, \quad (1)$$

where $\widetilde{S}$ denotes the union of significant patterns (node strings) on the two trees. The similarity score sim_p includes the distance associated with a pattern s, defined as

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

where $d(s)$ is the euclidean distance associated with a pattern s over T_i and T_j.

For a pattern $s \in T$, $P^T(s)$ 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. Because the conditional empirical probabilities are also parts of a pattern, we consider the conditional empirical probabilities $P^T(\sigma|s)$ when calculating the distance between two PSTs. Therefore, we sum $d(s)$ for all $s \in \widetilde{S}$ as the distance between two PSTs. Note that the distance between two PSTs is normalized by its maximal value, i.e., $2L_{max} + \sqrt{2}$. We take

2. sim_p is to be defined in Section 4.1.

3. In this work, we set the threshold to $\frac{1}{2}$, as suggested in [37], and leave the similarity threshold as the major control parameter because training the similarity threshold of two objects is easier than that of a group of objects.

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. With the definition of similarity score, two objects are similar to each other if their score is above a specified similarity threshold.

The GMPMine algorithm includes four steps. First, we extract the movement patterns from the location sequences by learning a PST for each object. Second, our algorithm constructs an undirected, unweighted similarity graph where similar objects share an edge between each other. We model the density of group relationship by the connectivity of a subgraph, which is also defined as the minimal cut of the subgraph. When the ratio of the connectivity to the size of the subgraph is higher than a threshold, the objects corresponding to the subgraph are identified as a group. Since the optimization of the graph partition problem is intractable in general, we bisect the similarity graph in the following way. We leverage the HCS cluster algorithm [37] to partition the graph and derive the location group information. Finally, we select a group PST T_g for each group in order to conserve the memory space by using the formula expressed as $T_g = arg\max_{T \in \overline{T}} \sum_{s \in \overline{S}} P^T(s)$, where $\overline{S}$ denotes sequences of a group of objects and $\overline{T}$ denotes their PSTs. Due to the page limit, we give an illustrative example of the GMPMine algorithm in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30.

4.2 The Cluster Ensembling (CE) Algorithm

In the previous section, each CH collects location data and generates local group results with the proposed GMPMine algorithm. Since the objects may pass through only partial sensor clusters and have different movement patterns in different clusters, the local grouping results may be inconsistent. For example, if objects in a sensor cluster walk close together across a canyon, it is reasonable to consider them a group. In contrast, objects scattered in grassland may not be identified as a group. Furthermore, in the case where a group of objects moves across the margin of a sensor cluster, it is difficult to find their group relationships. Therefore, we propose 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 all moving objects O that contains the most information about the local grouping results. We utilize NMI [38], [39] to evaluate the grouping quality. Let C denote the ensemble of the local grouping results, represented as $C = \{G_0, G_1, \ldots, G_K\}$, where K denotes the ensemble size. Our goal is to discover the ensembling result G' that contains the most information about C, i.e., $G' = arg\max_{G \in \widetilde{G}} \sum_{i=1}^{K} NMI(G_i, G)$, 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 the resource-constrained environments. To overcome this difficulty, the CE algorithm 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 $[0, 1]$ such that a finer-grained configuration of D achieves a better grouping quality but in a higher computation cost. Therefore, for a set of thresholds D, we rewrite our objective function as $G_{\delta'} = arg\max_{G_{\delta}, \delta \in D} \sum_{i=1}^{K} NMI(G_i, G_{\delta})$.

The algorithm includes three steps. First, we utilize Jaccard Similarity Coefficient [40] as the measure of the similarity for each pair of objects. Second, for each $\delta \in D$, we construct a graph where two objects share an edge if their Jaccard Similarity Coefficient is above δ. Our algorithm partitions the objects to generate a partitioning result G_{δ}. Third, we select the ensembling result $G_{\delta'}$. Because of space limitations, we only demonstrate the CE mining algorithm with an example in Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30. In the next section, we propose our compression algorithm that leverages the obtained group movement patterns.

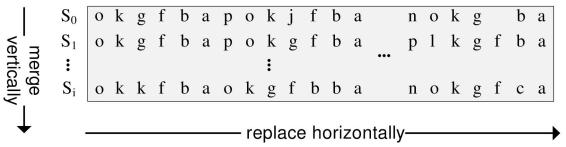


Fig. 4. Design of the two-phase and 2D compression algorithm.

5 Design of a Compression Algorithm with Group Movement Patterns

A WSN is composed of a large number of miniature sensor nodes that are deployed in a remote area for various applications, such as environmental monitoring or wildlife tracking. These sensor nodes are usually battery-powered and recharging a large number of them is difficult. Therefore, energy conservation is paramount among all design issues in WSNs [41], [42]. Because the target objects are moving, conserving energy in WSNs for tracking moving objects is more difficult than in WSNs that monitor immobile phenomena, such as humidity or vibrations. Hence, previous works, such as [43], [44], [45], [46], [47], [48], especially consider movement characteristics of moving objects in their designs to track objects efficiently. On the other hand, since transmission of data is one of the most energy expensive tasks in WSNs, data compression is utilized to reduce the amount of delivered data [1], [2], [3], [4], [5], [6], [49], [50], [51], [52]. Nevertheless, few of the above works have addressed the application-level semantics, i.e., the temporal-and-spatial correlations of a group of moving objects.

Therefore, to reduce the amount of delivered data, we propose the 2P2D algorithm which leverages the group movement patterns derived in Section 4 to compress the location sequences of moving objects elaborately. As shown in Fig. 4, the algorithm includes the sequence merge phase and the entropy reduction phase to compress location sequences vertically and horizontally. In the sequence merge phase, we propose the Merge algorithm to compress the location sequences of a group of moving objects. Since objects with similar movement patterns are identified as a group, their location sequences are similar. The Merge algorithm avoids redundant sending of their locations, and thus, reduces the overall sequence length. It combines the sequences of a group of moving objects by 1) trimming

multiple identical symbols at the same time interval into a single symbol or 2) choosing a qualified symbol to represent them when a tolerance of loss of accuracy is specified by the application. Therefore, the algorithm trims and prunes more items when the group size is larger and the group relationships are more distinct. Besides, in the case that only the location center of a group of objects is of interest, our approach can find the aggregated value in the phase, instead of transmitting all location sequences back to the sink for postprocessing.

In the entropy reduction phase, we propose the Replace algorithm that utilizes the group movement patterns as the prediction model to further compress the merged sequence. The Replace algorithm guarantees the reduction of a sequence's entropy, and consequently, improves compressibility without loss of information. Specifically, we define a new problem of minimizing the entropy of a sequence as the HIR problem. To reduce the entropy of a location sequence, we explore Shannon's theorem [17] to derive three replacement rules, based on which the Replace algorithm reduces the entropy efficiently. Also, we prove that the Replace algorithm obtains the optimal solution of the HIR problem as Theorem 1. In addition, since the objects may enter and leave a sensor cluster multiple times during a batch period and a group of objects may enter and leave a cluster at slightly different times, we discuss the segmentation and alignment problems in Section 5.3. Table 1 summaries the notations.

5.1 Sequence Merge Phase

In the application of tracking wild animals, multiple moving objects may have group relationships and share similar trajectories. In this case, transmitting their location data separately leads to redundancy. Therefore, in this section, we concentrate on the problem of compressing multiple similar sequences of a group of moving objects.

Consider an illustrative example in Fig. 5a, where three location sequences S_0, S_1, and S_2 represent the trajectories of a group of three moving objects. Items with the same index belong to a column, and a column containing identical symbols is called an S-column; otherwise, the column is called a D-column. Since sending the items in an S-column individually causes redundancy, our basic idea of compressing multiple sequences is to trim the items in an

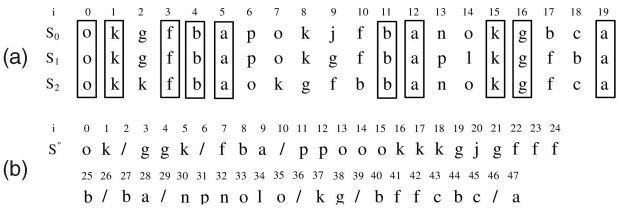


Fig. 5. An example of the Merge algorithm. (a) Three sequences with high similarity. (b) The merged sequence S''.

S-column into a single symbol. Specifically, given a group of n sequences, the items of an S-column are replaced by a single symbol, whereas the items of a D-column are wrapped up between two $'/'$ symbols. Finally, our algorithm generates a merged sequence containing the same information of the original sequences. In decompressing from the merged sequence, while symbol $'/'$ is encountered, the items after it are output until the next $'/'$ symbol. Otherwise, for each item, we repeat it n times to generate the original sequences. Fig. 5b shows the merged sequence S''whose length is decreased from 60 items to 48 items such that 12 items are conserved. The example pointed out that our approach can reduce the amount of data without loss of information. Moreover, when there are more S-columns, our approach can bring more benefit.

When a little loss in accuracy is tolerant, representing items in a D-column by an qualified symbol to generate more S-columns can improve the compressibility. We regulate the accuracy by an error bound, defined as the maximal hop count between the real and reported locations of an object. For example, replacing items 2 and 6 of S_2 by $'g'$ and $'p'$, respectively, creates two more S-columns, and thus, results in a shorter merged sequence with 42 items. To select a representative symbol for a D-column, we includes a selection criterion to minimize the average deviation between the real locations and reported locations for a group of objects at each time interval as follows:

Selection criterion. *The maximal distance between the reported location and the real location is below a specified error bound eb, i.e., when the ith column is a D-column,* $|S_j[i] - \sigma| \leq eb$ *must hold for* $0 \leq j < n$, *where* n *is the number of sequences.*

TABLE 1
Description of the Notations

Notation	Description
Σ	The alphabet of sensor IDs.
σ	A symbol in Σ.
S	A location sequence.
$'.'$	The hit symbol that is used to replace a predictable item.
$S_i[j]$	j-th item of S_i.
$S_i[j..k]$	A subsequence containing j-th, $(j+1)$-th,... k-th items of S_i.
$n(\sigma)$	The number of items of σ in S.
$n_{hit}(\sigma)$	The number of predictable items of σ in S.
$\hat{s}$	The sub-set of Σ, where $n_{hit}(\sigma) > 0$ for every $\sigma \in \hat{s}$.
p_i	The occurrence probability of σ_i corresponding to S.

```
Algorithm: Merge
Input: a group of sequences { Si| 0 ≤ i< n} with length L
       an error bound eb
Output: the merged sequence S"
0.    initialize ps, S"
1.    dc_start= 0
2.    for 0 ≤ j < L
3.        σ = null
4.        if is_S-column ( Si[j] ,0 ≤ i < n) then
5.            σ = So[j]
6.        else if eb > 0 then
7.            σ = getRS ({Si[j], 0 ≤ i < n}, eb)
8.        if σ == null then
9.            if dc _start== 0 then
10.               append(S",'/')
11.               dc _start= 1
12.           for 0 ≤ i < n
13.               append (S", Si[j])
14.       else
15.           if dc _start == 1 then
16.               append(S",'/')
17.               dc _start= 0
18.           append (S", σ )
19.   return S"
```

Fig. 6. The Merge algorithm.

Therefore, to compress the location sequences for a group of moving objects, we propose the Merge algorithm shown in Fig. 6. The input of the algorithm contains a group of sequences $\{S_i|0 \leq i < n\}$ and an error bound eb, while the output is a merged sequence that represents the group of sequences. Specifically, the Merge algorithm processes the sequences in a columnwise way. For each column, the algorithm first checks whether it is an S-column. For an S-column, it retains the value of the items as Lines 4-5. Otherwise, while an error bound $eb > 0$ is specified, a representative symbol is selected according to the selection criterion as Line 7. If a qualified symbol exists to represent the column, the algorithm outputs it as Lines 15-18. Otherwise, the items in the column are retained and wrapped by a pair of "/" as Lines 9-13. The process repeats until all columns are examined. Afterward, the merged sequence S'' is generated.

Following the example shown in Fig. 5a, with a specified error bound eb as 1, the Merge algorithm generates the solution, as shown in Fig. 7; the merged sequence contains only 20 items, i.e., 40 items are curtailed.

5.2 Entropy Reduction Phase

In the entropy reduction phase, we propose the Replace algorithm to minimize the entropy of the merged sequence obtained in the sequence merge phase. Since data with lower entropy require fewer bits for storage and transmission [17], we replace some items to reduce the entropy without loss of information. The object movement patterns discovered by our distributed mining algorithm enable us to find the replaceable items and facilitate the selection of items in our compression algorithm. In this section, we first introduce and define the HIR problem, and then, explore the properties of Shannon's entropy to solve the HIR problem. We extend the concentration property for entropy reduction and discuss the benefits of replacing multiple symbols simultaneously. We derive three replacement rules for the HIR problem and prove that the entropy of the obtained solution is minimized.

5.2.1 Three Derived Replacement Rules

Let $P = \{p_0, p_1, \ldots, p_{|\Sigma|-1}\}$ denote the probability distribution of the alphabet Σ corresponding to a location sequence S, where p_i is the occurrence probability of σ_i in Σ, defined as $p_i = \frac{|\{s_j|s_j=\sigma_i, 0 \leq j < L\}|}{L}$, and $L = |S|$. According to Shannon's source coding theorem, the optimal code length for symbol σ_i is $\log_2 p_i$, which is also called the information content of σ_i. Information entropy (or Shannon's entropy) is thereby defined as the overall sum of the information content over Σ,

$$e(S) = e(p_0, p_1, \ldots, p_{|\Sigma|-1}) = \sum_{0 \leq i < |\Sigma|} -p_i \times \log_2 p_i. \quad (2)$$

Shannon's entropy represents the optimal average code length in data compression, where the length of a symbol's codeword is proportional to its information content [17].

A property of Shannon's entropy is that the entropy is the maximum, while all probabilities are of the same value. Thus, in the case without considering the regularity in the movements of objects, the occurrence probabilities of symbols are assumed to be uniform such that the entropy of a location sequence as well as the compression ratio is fixed and dependent on the size of a sensor cluster, e.g., for a location sequence with length D collected in a sensor cluster of 16 nodes, the entropy of the sequence is $e(\frac{1}{16}, \frac{1}{16}, \ldots, \frac{1}{16}) = 4$. Consequently, $4D$ bits are needed to represent the location sequence. Nevertheless, since the movements of a moving object are of some regularity, the occurrence probabilities of symbols are probably skewed and the entropy is lower. For example, if two probabilities become $\frac{1}{32}$ and $\frac{3}{32}$, the entropy is reduced to 3.97. Thus, instead of encoding the sequence using 4 bits per symbol, only $3.97D$ bits are required for the same information. This example points out that when a moving object exhibits some degree of regularity in their movements, the skewness of these probabilities lowers the entropy. Seeing that data with lower entropy require fewer bits to represent the same information, reducing the entropy thereby benefits for data compression and, by extension, storage and transmission.

Motivated by the above observation, we design the Replace algorithm to reduce the entropy of a location sequence. Our algorithm imposes the hit symbols on the location sequence to increase the skewness. Specifically, the algorithm uses the group movement patterns built in both the transmitter (CH) and the receiver (sink) as the prediction model to decide whether an item of a sequence is predictable. A CH replaces the predictable items each with a hit symbol to reduce the location sequence's entropy when compressing it. After receiving the compressed sequence, the sink node decompresses it and substitutes every hit symbol with the original symbol by the identical prediction model, and no information loss occurs.

Here, an item s_i of a sequence S is predictable item if $predict\_next(T, S[0..i-1])$ is the same value as s_i. A symbol is a predictable symbol once an item of the symbol is predictable. For ease of explanation, we use a $taglst$[4] to

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S"	o	k	g	f	b	a	p	o	k	f	f	b	a	o	k	k	g	f	b	a

Fig. 7. An example of the Merge algorithm: the merged sequence with $eb = 1$.

4. A $taglst$ associated with a sequence S is a sequence of 0s and 1s obtained as follows: For those predictable items in S, the corresponding items in $taglst$ are set as 1. Otherwise, their values in $taglst$ are 0.

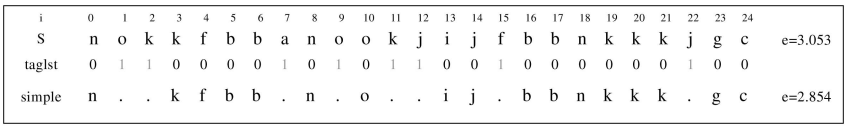


Fig. 8. An example of the simple method. The first row represents the index, and the second row is the location sequence. The third row is the taglst which represents the prediction results. The last row is the result of the simple approach. Note that items 1, 2, 7, 9, 11, 12, 15, and 22 are predictable items such that the set of predictable symbols is $\hat{s} = \{k, a, j, f, o\}$. In the example, items 2, 9, and 10 are items of $'o'$ such that the number of items of symbol $'o'$ is $n('o') = 3$, whereas items 2 and 9 are the predictable items of $'o'$, and the number of predictable items of symbol $'o'$ is $n_{hit}('o') = 2$.

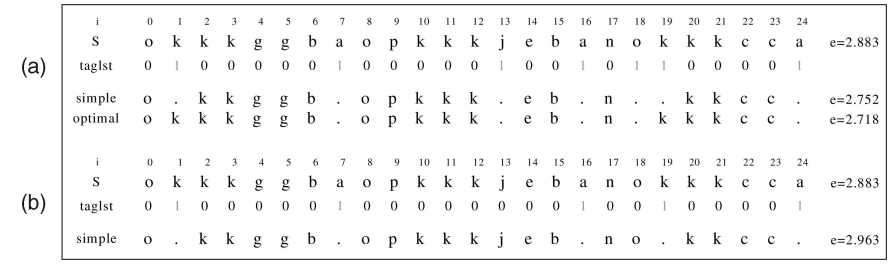


Fig. 9. Problems of the simple approach.

express whether each item of S is predictable in the following sections. Consider the illustrative example shown in Fig. 8, the probability distribution of Σ corresponding to S is $P = \{0.04, 0.16, 0.04, 0, 0, 0.08, 0.04, 0, 0.04, 0.12, 0.24, 0, 0, 0.12, 0.12, 0\}$, and items 1, 2, 7, 9, 11, 12, 15, and 22 are predictable items. To reduce the entropy of the sequence, a simple approach is to replace each of the predictable items with the hit symbol and obtain an intermediate sequence $S' =''n..kfbb.n.o..ij.bbnkkk.gc''$. Compared with the original sequence S with entropy 3.053, the entropy of S' is reduced to 2.854. Encoding S and S' by the Huffman coding technique, the lengths of the output bit streams are 77 and 73 bits, respectively, i.e., 5 bits are conserved by the simple approach.

However, the above simple approach does not always minimize the entropy. Consider the example shown in Fig. 9a, an intermediate sequence with items 1 and 19 unreplaced has lower entropy than that generated by the simple approach. For the example shown in Fig. 9b, the simple approach even increases the entropy from 2.883 to 2.963.

We define the above problem as the HIR problem and formulate it as follows:

Definition 3 (HIR problem). *Given a sequence $S = \{s_i | s_i \in \Sigma, 0 \leq i < L\}$ and a taglst, an intermediate sequence is a generation of S, denoted by $S' = \{s'_i | 0 \leq i < L\}$, where s'_i is equal to s_i if $taglst[i] = 0$. Otherwise, s'_i is equal to s_i or $'.'$. The HIR problem is to find the intermediate sequence S' such that the entropy of S' is minimal for all possible intermediate sequences.*

A brute-force method to the HIR problem is to enumerate all possible intermediate sequences to find the optimal solution. However, this brute-force approach is not scalable, especially when the number of the predictable items is large. Therefore, to solve the HIR problem, we explore properties of Shannon's entropy to derive three replacement rules that our Replace algorithm leverages to obtain the optimal solution. Here, we list five most relevant properties to explain the replacement rules. The first four properties can be obtained from [17], [53], [54], while the fifth, called the strong concentration property, is derived and proved in the paper.[5]

Property 1 (Expansibility). *Adding a probability with a value of zero does not change the entropy, i.e., $e(p_0, p_1, \ldots, p_{|\Sigma|-1}, p_{|\Sigma|})$ is identical to $e(p_0, p_1, \ldots, p_{|\Sigma|-1})$ when $p_{|\Sigma|}$ is zero.*

According to Property 1, we add a new symbol $'.'$ to Σ without affecting the entropy and denote its probability as p_{16} such that P is $\{p_0, p_1, \ldots, p_{15}, p_{16} = 0\}$.

Property 2 (Symmetry). *Any permutation of the probability values does not change to the entropy. For example, $e(0.1, 0.4, 0.5)$ is identical to $e(0.4, 0.5, 0.1)$.*

Property 3 (Accumulation). *Moving all the value from one probability to another such that the former can be thought of as being eliminated decreases the entropy, i.e., $e(p_0, p_1, \ldots, 0, \ldots, p_i + p_j, \ldots, p_{|\Sigma|-1})$ is equal or less than $e(p_0, p_1, \ldots, p_i, \ldots, p_j, \ldots, p_{|\Sigma|-1})$.*

With Properties 2 and 3, if all the items of symbol σ are predictable, i.e., $n(\sigma) = n_{hit}(\sigma)$, replacing all the items of σ by $'.'$ will not affect the entropy. If there are multiple symbols having $n(\sigma) = n_{hit}(\sigma)$, replacing all the items of these symbols can reduce the entropy. Thus, we derive the first replacement rule—**the accumulation rule: Replace all items of symbol σ in $\hat{s}$, where $n(\sigma) = n_{hit}(\sigma)$.**

Property 4 (Concentration). *For two probabilities, moving a value from the lower probability to the higher probability decreases the entropy, i.e., if $p_i \leq p_j$, for $0 < \Delta p \leq p_i$, $e(p_0, p_1, \ldots, p_i - \Delta p, \ldots, p_j + \Delta p, \ldots, p_{|\Sigma|-1})$ is less than $e(p_0, p_1, \ldots, p_i, \ldots, p_j, \ldots, p_{|\Sigma|-1})$.*

Property 5 (Strong concentration). *For two probabilities, moving a value that is larger than the difference of the two probabilities from the higher probability to the lower one decreases the entropy, i.e., if $p_i > p_j$, for $p_i - p_j < \Delta p \leq p_i$,*

5. Due to the page limit, the proofs of the strong concentration rule and Lemmas 1-4 are given in Appendices C and D, respectively, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30.

$e(p_0,p_1,\ldots,p_i-\Delta p,\ldots,p_j+\Delta p,\ldots,p_{|\Sigma|-1})$ *is less than* $e(p_0,p_1,\ldots,p_i,\ldots,p_j,\ldots,p_{|\Sigma|-1})$.

According to Properties 4 and 5, we conclude that if the difference of two probabilities increases, the entropy decreases. When the conditions conform to Properties 4 and 5, we further prove that the entropy is monotonically reduced as Δp increases such that replacing all predictable items of a qualified symbol reduces the entropy mostly as Lemmas 1 and 2.

Definition 4 (Concentration function). *For a probability distribution* $P=\{p_0, p_1,\ldots,p_{|\Sigma|-1}\}$, *we define the concentration function of k probabilities* $p_{i_1}, p_{i_2},\ldots,p_{i_k}$, *and* p_j *as*

$$f_{i_1 i_2 \ldots i_k j}(x_1,x_2,\ldots,x_k) = e\Big(p_0,\ldots,p_{i_1}-x_1,\ldots,p_{i_2} - x_2,\ldots,p_{i_k}-x_k,\ldots,p_j + \sum_{i=1}^{k} x_i,\ldots,p_{|\sum|-1}\Big).$$

Lemma 1. *If* $p_i \le p_j$, *for any* x *such that* $0 \le x \le p_i$, *the concentration function of* p_i *and* p_j *is monotonic decreasing.*

Lemma 2. *If* $p_i > p_j$, *for any* x *such that* $p_i - p_j \le x \le p_i$, *the concentration function* $f_{ij}(x)$ *of* p_i *and* p_j *is monotonic decreasing.*

Accordingly, we derive the second replacement rule—**the concentration rule: Replace all predictable items of symbol** σ **in** $\hat{s}$**, where** $n(\sigma) \le n('.')$ **or** $n_{hit}(\sigma) > n(\sigma) - n('.')$.

As an extension of the above properties, we also explore the entropy variation, while predictable items of *multiple* symbols are replaced simultaneously. Let $\hat{s}'$ denote a subset of $\hat{s}$, where $|\hat{s}'| \ge 2$. We prove that if replacing predictable symbols in $\hat{s}'$ can minimize the entropy corresponding to symbols in $\hat{s}'$, the number of hit symbols after the replacement must be larger than the number of the predictable symbols after the replacement as Lemma 3. To investigate whether the converse statement exists, we conduct an experiment in a brute-force way. However, the experimental results show that even under the condition that $n('.') + \sum_{\forall \sigma' \in \hat{s}'} n_{hit}(\sigma') \ge n(\sigma) - n_{hit}(\sigma)$ for every σ in $\hat{s}'$, replacing predictable items of the symbols in $\hat{s}'$ does not guarantee the reduction of the entropy. Therefore, we compare the difference of the entropy before and after replacing symbols in $\hat{s}'$ as

$$\begin{aligned} gain(\hat{S}') = &- \sum_{\forall \sigma \in \hat{s}'} p_\sigma \times \log_2 \frac{p_\sigma}{p_\sigma - \Delta p_\sigma} - \Delta_{p_\sigma} \\ &\times \log_2(p_\sigma - \Delta_{p_\sigma}) \\ &- p_{'.'} \times \log_2\left(\frac{p_{'.'}}{p'.' + \sum_{\forall \sigma \in \hat{s}'} \Delta_{p-\sigma}}\right) \\ &+ \sum_{\forall \sigma \in \hat{s}'} \Delta p_\sigma \times \log_2\left(p_{'.'} + \sum_{\forall \sigma \in \hat{s}'} \Delta_{p-\sigma}\right). \end{aligned}$$

In addition, we also prove that once replacing partial predictable items of symbols in $\hat{s}'$ reduces entropy, replacing all predictable items of these symbols reduces the entropy mostly since the entropy decreases monotonically as Lemma 4. We thereby derive the third replacement rule—**the multiple symbol rule: Replace all of the predictable items of every symbol in** $\hat{s}'$ **if** $gain(\hat{s}') > 0$.

Lemma 3. *If exist* $0 \le a_n \le p_{i_n}$, $n=1,\ldots,k$, *such that* $f_{i_1 i_2 \cdots i_k j}(a_1,a_2,\ldots,a_k)$ *is minimal, we have* $p_j + \sum_{i=1}^{k} a_i \ge p_{i_n} - a_n$, $n=1,\ldots,k$.

Lemma 4. *If exist* x_{min_1}, $x_{min_2},\ldots,$ *and* x_{min_k} *such that* $p_j + \sum_{i=1}^{k} x_{min_i} > p_{i_n} - x_{min_n}$ *for* $n=1,2,\ldots,k, f_{i_1 i_2 \cdots i_k j}(x_1,x_2,\ldots,x_k)$ *is monotonically decreasing for* $x_{min_n} \le x_n \le p_{i_n}$, $n=1,\ldots,k$.

5.2.2 The Replace Algorithm

Based on the observations described in the previous section, we propose the Replace algorithm that leverages the three replacement rules to obtain the optimal solution for the HIR problem. Our algorithm examines the predictable symbols on their statistics, which include the number of items and the number of predictable items of each predictable symbol. The algorithm first replaces the qualified symbols according to the accumulation rule. Afterward, since the concentration rule and the multiple symbol rule are related to $n('.')$, which is increased after every replacement, the algorithm iteratively replaces the qualified symbols according to the two rules until all qualified symbols are replaced. The algorithm thereby replaces qualified symbols and reduces the entropy toward the optimum gradually. Compared with the brute-force method that enumerates all possible intermediate sequences for the optimum in exponential complexity, the Replace algorithm that leverages the derived rules to obtain the optimal solution in $O(L)$ time[6] is more scalable and efficient. We prove that the Replace algorithm guarantees to reduce the entropy monotonically and obtains the optimal solution of the HIR problem as Theorem 1. Next, we detail the replace algorithm and demonstrate the algorithm by an illustrative example.

Theorem 1.[7] *The Replace algorithm obtains the optimal solution of the HIR problem.*

Fig. 10 shows the Replace algorithm. The input includes a location sequence S and a predictor T_g, while the output, denoted by S', is a sequence in which qualified items are replaced by $'.'$. Initially, Lines 3-9 of the algorithm find the set of predictable symbols together their statistics. Then, it exams the statistics of the predictable symbols according to the three replacement rules as follows: First, according to the accumulation rule, it replaces qualified symbols in one scan of the predictable symbols as Lines 10-14. Next, the algorithm iteratively exams for the concentration and the multiple symbol rules by two loops. The first loop from Line 16 to Line 22 is for the concentration, whereas the second loop from Line 25 to Line 36 is for the multiple symbol rule. In our design, since finding a combination of predictable symbols to make $gain(\hat{s}') > 0$ hold is more costly, the algorithm is prone to replace symbols with the

6. The complexity analysis is given in Appendix E, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30.

7. The proof of Theorem 1 is given in Appendix F, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.30.

```
Algorithm: Replace
Input: a sequence: S
       a predictor: T_g
Output: an intermediate sequence: S'
0.    ŝ = ϕ
1.    n(".") = 0
2.    taglst = get_taglst(S, T_g)
3.    for 0 ≤ i < |S|-1                    /* getting statistics of S */
4.        σ = S[i]
5.        n(σ) ++
6.        if taglst[i] == 1 then
7.            n_hit(σ) = n_hit(σ) + 1
8.            if σ ∉ ŝ then
9.                add σ to ŝ
10.   for ∀ σ ∈ ŝ                          /* the accumulation rule */
11.       if n(σ) == n_hit(σ) then
12.           replaceHitItems(S, taglst, σ)
13.           n(".") = n(".") + n(σ)
14.           remove σ from ŝ
15.   while ŝ ≠ ϕ
16.       do                               /* the concentration rule */
17.           for ∀ σ ∈ ŝ
18.               if n(σ) < n(".") or n_hit(σ) > n(σ) - n(".") then
19.                   replaceHitItems(S, taglst, σ)
20.                   n(".") = n(".") + n_hit(σ)
21.                   remove σ from ŝ
22.       until n(".") is no more increased
23.       m = 2                            /* the multiple symbol rule */
24.       ŝ' = get a combination of m symbols from ŝ
25.       while ŝ' ≠ ϕ
26.           if gain(S, ŝ') > 0 then
27.               for ∀ σ ∈ ŝ'
28.                   replaceHitItems(S, taglst, σ)
29.                   n(".") = n(".") + n_hit(σ)
30.                   remove σ from ŝ
31.               break  /* once n(".") is changed, exit the while loop */
32.           else
33.               ŝ' = get next combination of m symbols from ŝ
34.               if ŝ' == ϕ and m < |ŝ| then
35.                   m++
36.                   ŝ' = get a combination of m symbols from ŝ
37.   return S'
```

Fig. 10. The Replace algorithm.

concentration rule. Specifically, after a scan of predictable symbols for the second rule as Lines 17-23, the algorithm search for a combination of symbols in $\hat{s}$ to make the condition of the multiple symbol rule hold as Lines 26-36; it starts with a combination of two symbols, i.e., $m = 2$. Once a combination $\hat{s}'$ makes $gain(\hat{s}') > 0$ hold, the enumeration procedure stops as Line 31 and the algorithm goes back to the first loop. Otherwise, after an exhaustive search for any combination of m symbols, it goes on examining the combinations of $m + 1$ symbols. The process repeats until $\hat{s}'$ contains all of the symbols in $\hat{s}$.

In the following, we explain our Replace algorithm with an illustrative example. Given a sequence $S = "nokkfbbanookjijfbbnkkkjgc"$ with entropy 3.053, as shown in Fig. 11, our algorithm generates the statistic table and $taglst$ shown in Fig. 11a. In this example, $\hat{s} = \{k, j, o, a, f\}$, and the numbers of items of the symbols are 6, 3, 3, 1, and 2, whereas the numbers of predictable items of the symbols are 2, 2, 2, 1, and 1, respectively. First, according to the accumulation rule, the predictable items of $'a'$ are replaced due to $n('a')$ being equal to $n_{hit}('a')$ (Lines 10-14). After that, the statistic table is updated, as shown in Fig. 11b. Second, according to the multiple symbol rule, we replace the predictable items of $'j'$ and $'o'$ simultaneously such that the entropy of S' is reduced to 2.969. Next, because $n('f')$ is less than $n('.')$, the predictable items of $'f'$ are replaced according to the concentration rule (Lines 17-23), then the entropy of S' is reduced to 2.893. Finally, since $n('k')$ is equal to $n('.')$ and $n_{hit}('k')$ is greater than $n('k')$ minus $('.')$, the predictable items of symbol $'k'$ are replaced according to the concentration rule. Finally, no other candidate is available, and our algorithm outputs S' with entropy 2.854. In this example, all predictable items are replaced to minimize the entropy.

In addition, for the example shown in Fig. 5, the Replace algorithm reduces S_0, S_1, and S_2's entropies from 3.171, 2.933, and 2.871 to 2.458, 2.828, and 2.664, respectively, and encoding S_0', S_1', and S_2' reduces the sum of output bitstreams from 181 to 161 bits. On the other hand, when the specified error bound eb is 0 and 1, by fully utilizing the group movement patterns, the 2P2D algorithm reduces the total data size to 153 and 47 bits, respectively; hence, 15.5 and 74 percent of the data volume are saved, respectively.

5.3 Segmentation, Alignment, and Packaging

In an online update approach, sensor nodes are assigned a tracking task to update the sink with the location of moving objects at every tracking interval. In contrast to the online

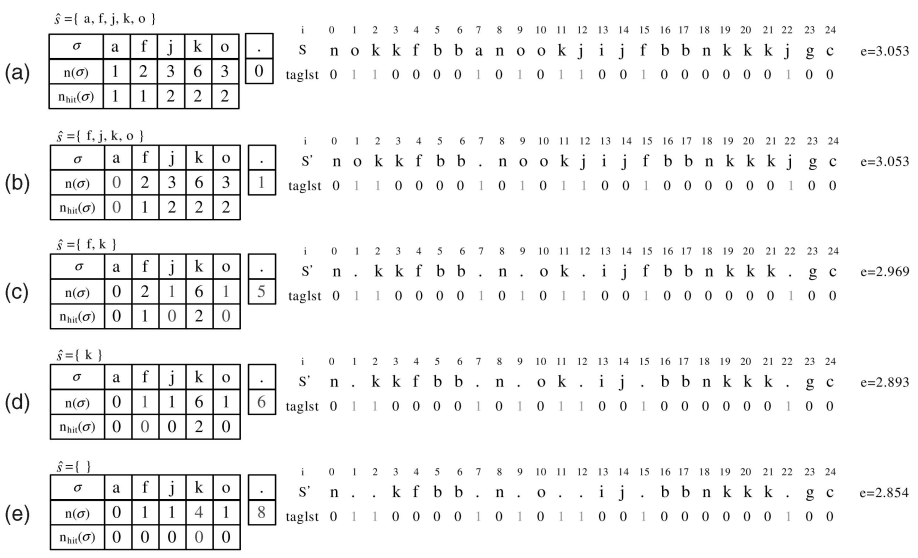


Fig. 11. An example of the Replace algorithm.

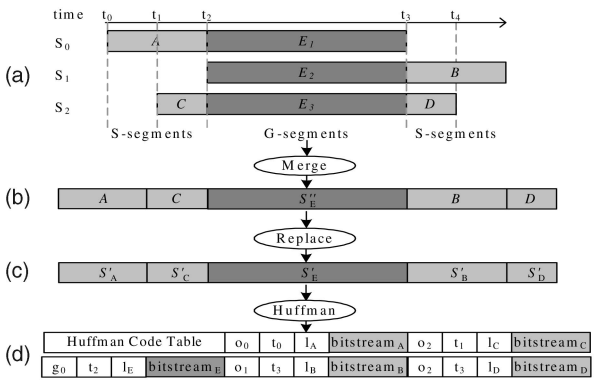


Fig. 12. An example of constructing an update packet. (a) Three sequences aligned in time domain. (G-segments: E_1, E_2, and E_3, S-segments: A, B , C, and D.) (b) Combining G-segments by the Merge algorithm. (c) Replacing the predictable items by the Replace algorithm. (d) Compressing and packing to generate the payload of the update packet.

approach, the CHs in our batch-based approach accumulate a large volume of location data for a batch period before compressing and transmitting it to the sink; and the location update process repeats from batch to batch. In real-world tracking scenarios, slight irregularities of the movements of a group of moving objects may exist in the microcosmic view. Specifically, a group of objects may enter a sensor cluster at slightly different times and stay in a sensor cluster for slightly different periods, which lead to the alignment problem among the location sequences. Moreover, since the trajectories of moving objects may span multiple sensor clusters, and the objects may enter and leave a cluster multiple times during a batch period, a location sequence may comprise multiple segments, each of which is a trajectory that is continuous in time domain. To deal with the alignment and segmentation problems, we partition location sequences into segments, and then, compress and package them into one update packet.

Consider a group of three sequences shown in Fig. 12a, the segments E_1, E_2, and E_3 are aligned and named G-segments, whereas segments A, B, C, and D are named S-segments. Figs. 12b, 12c, and 12d show an illustrative example to construct the frame for the three sequences. First, the Merge algorithm combines E_1, E_2, and E_3 to generate an intermediate sequence S''_E. Next, S''_E together with A, B, C, and D is viewed as a sequence and processed by the Replace algorithm to generate an intermediate sequence S', which comprises S'_A, S'_B, S'_C, S'_D, and S'_E. Finally, intermediate sequence S' is compressed and packed.

For a batch period of D tracking intervals, the location data of a group of n objects are aggregated in one packet such that $(n \times D - 1)$ packet headers are eliminated. The payload may comprise multiple G-segments or S-segments, each of which includes a beginning time stamp (a bits), a sequence of consequent locations (b bits for each), an object or group ID (c bits), and a field representing the length of a segment (l bits). Therefore, the payload size is calculated as $\sum_i (a + D_i \times b + c + l)$, where D_i is the length of ith segment. By exploiting the correlations in the location data, we can further compress the location data and reduce the amount of data to $H + \sum_i (a + c + l) + n \times D \times b \times \frac{1}{r}$, where r denotes the compression ratio of our compression algorithm and H denotes the data size of the packet header.

As for the online update approach, when a sensor node detects an object of interest, it sends an update packet upward to the sink. The payload of a packet includes time stamp, location, and object ID such that the packet size is $H + a + b + c$. Some approaches, such as [55], employ techniques like location prediction to reduce the number of transmitted update packets. For D tracking intervals, the amount of data for tracking n objects is $D \times (H + a + b + c) \times (1 - p) \times n$, where p is the prediction hit rate.

Therefore, the group size, the number of segments, and the compress ratio are important factors that influence the performance of the batch-based approach. In the next section, we conduct experiments to evaluate the performance of our design.

6 Experiment and Analysis

We implement an event-driven simulator in C++ with SIM [56] to evaluate the performance of our design. To the best of our knowledge, no research work has been dedicated to discovering application-level semantic for location data compression. We compare our batch-based approach with an online approach for the overall system performance evaluation and study the impact of the group size (n), as well as the group dispersion radius (GDR), the batch period (D), and the error bound of accuracy (eb). We also compare our Replace algorithm with Huffman encoding technique to show its effectiveness. Since there is no related work that finds real location data of group moving objects, we generate the location data, i.e., the coordinates (x, y), with the Reference Point Group Mobility Model [57] for a group of objects moving in a two-layer tracking network with 256 nodes. A location-dependent mobility model [58] 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 followers and the leader. We utilize the GDR to control the dispersion degree of the objects. Smaller GDR implies stronger group relationships, i.e., objects are closer together. The speed of each object is 1 node per time unit, and the tracking interval is 0.5 time unit. In addition, the starting point and the furthest point reached by the leader object are randomly selected, and the movement range of a group of objects is the euclidean distance between the two points. Note that we take the group leader as a virtual object to control the roaming behavior of a group of moving objects and exclude it in calculating the data traffic.

In the following experiments, the default values of n, D, d, and eb are 5, 1,000, 6, and 0. The data sizes of object (or group) ID, location ID, time stamp, and packet header are 1, 1, 1, and 4 bytes, respectively. We set the PST parameters $(L_{max}, P_{min}, \alpha, \gamma_{min}, r) = (5, 0.01, 0, 0.0001, 1.2)$ empirically in learning the movement patterns. Moreover, we use the amount of data in kilobyte (KB) and compression ratio (r) as the evaluation metric, where the compression ratio is

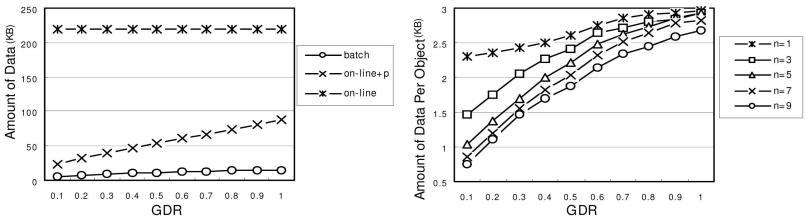


Fig. 13. Performance comparison of the batch-based and online update approaches and the impact of the group size. (a) Comparison of the batch-based and online approaches. (b) Impact of group size.

defined as the ratio between the uncompressed data size and the compressed data size, i.e., $r = \frac{\text{uncompressed data size}}{\text{compressed data size}}$.

First, we compare the amount of data of our batch-based approach (**batch**) with that of an online update approach (**online**). In addition, some approaches, such as [55], employ techniques like location prediction to reduce the number of transmitted update packets. We use the discovered movement patterns as the prediction model for prediction in the online update approach (**online**$+p$). Fig. 13a shows that our batch-based approach outperforms the online approach with and without prediction. The amount of data of our batch-based approach is relatively low and stable as the GDR increases. Compared with the online approach, the compression ratios of our batch approach and the online approach with prediction are about 15.0 and 2.5 as $GDR = 1$.

Next, our compression algorithm utilizes the group relationships to reduce the data size. Fig. 13b shows the impact of the group size. The amount of data per object decreases as the group size increases. Compared with carrying the location data for a single object by an individual packet, our batch-based approach aggregates and compresses packets of multiple objects such that the amount of data decreases as the group size increases. Moreover, our algorithm achieves high compression ratio in two ways. First, while more sequences that are similar or sequences that are more similar are compressed simultaneously, the Merge algorithm achieves higher compression ratio. Second, with the regularity in the movements of a group of objects, the Replace algorithm minimizes the entropy which also leads to higher compression ratio. Note that we use the GDR to control the group dispersion range of the input workload. The leader object's movement path together with the GDR sets up a spacious area where the member objects are randomly distributed. Therefore, a larger GDR implies that the location sequences have higher entropy, which degrades both the prediction hit rate and the compression ratio. Therefore, larger group size and smaller GDR result in higher compression ratio.

Fig. 14a shows the impact of the batch period (D). The amount of data decreases as the batch period increases. Since more packets are aggregated and more data are compressed for a longer batch period, our batch-based approach reduces both the data volume of packet headers and the location data.

Since the accuracy of sensor networks is inherently limited, allowing approximation of sensors' readings or tolerating a loss of accuracy is a compromise between data accuracy and energy conservation. We study the impact of accuracy on the amount of data. Fig. 14b shows that by tolerating a loss of accuracy with eb varying from 1 to 3, the amount of data decreases. As $GDR = 1$, the compression ratio r of $eb = 3$ is about 21.2; while the compression ratio r of $eb = 0$ is about 15.0.

We study the effectiveness of the Replace algorithm by comparing the compression ratios of the Huffman encoding with and without our Replace algorithm. As GDR varies from 0.1 to 1, Fig. 15a shows the compression ratios of the Huffman encoding with and without our Replace algorithm; while Fig. 15b shows the prediction hit rate. Compared with Huffman, our Replace algorithm achieves higher compression ratio, e.g., the compression ratio of our approach is about 4, while that of Huffman is about 2.65 as $GDR = 0.1$. From Figs. 15a and 15b, we show that the compression ratio that the Replace algorithm achieves reduces as the prediction hit rate. As the prediction hit rate is about 0.6, the compression ratio of our design is about 2.7 that is higher than 2.3 of Huffman.

7 Conclusions

In this work, we exploit the characteristics of group movements to discover the information about groups of moving objects in tracking applications. We propose a distributed mining algorithm, which consists of a local GMPMine

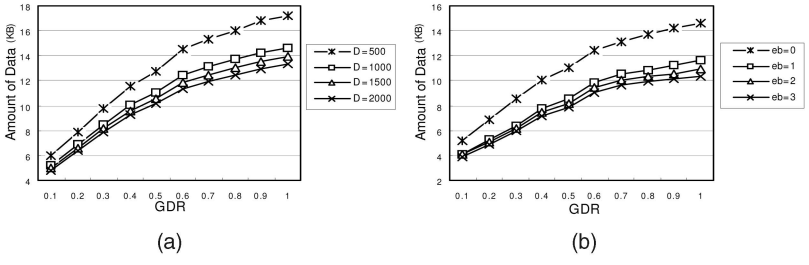


Fig. 14. Impacts of the batch period and accuracy. (a) Impact of batch period. (b) Impact of accuracy.

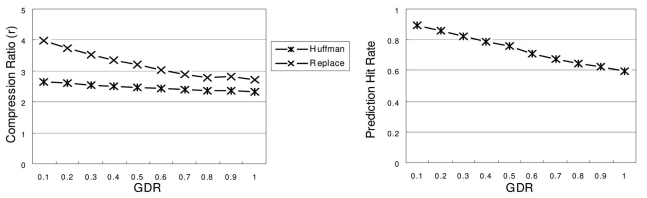


Fig. 15. Effectiveness of the Replace algorithm. (a) Compression ratio versus GDR. (b) Prediction hit rate versus GDR.

algorithm and a CE algorithm, to discover group movement patterns. With the discovered information, we devise the 2P2D algorithm, which comprises a sequence merge phase and an entropy reduction phase. In the sequence merge phase, we propose the Merge algorithm to merge the location sequences of a group of moving objects with the goal of reducing the overall sequence length. In the entropy reduction phase, we formulate the HIR problem and propose a Replace algorithm to tackle the HIR problem. In addition, we devise and prove three replacement rules, with which the Replace algorithm obtains the optimal solution of HIR efficiently. Our experimental results show that the proposed compression algorithm effectively reduces the amount of delivered data and enhances compressibility and, by extension, reduces the energy consumption expense for data transmission in WSNs.

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] S.S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed Compression in a Dense Microsensor Network," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 51-60, Mar. 2002.

[2] A. Scaglione and S.D. Servetto, "On the Interdependence of Routing and Data Compression in Multi-Hop Sensor Networks," *Proc. Eighth Ann. Int'l Conf. Mobile Computing and Networking,* pp. 140-147, 2002.

[3] N. Meratnia and R.A. de By, "A New Perspective on Trajectory Compression Techniques," *Proc. ISPRS Commission II and IV, WG II/5, II/6, IV/1 and IV/2 Joint Workshop Spatial, Temporal and Multi-Dimensional Data Modelling and Analysis,* Oct. 2003.

[4] S. Baek, G. de Veciana, and X. Su, "Minimizing Energy Consumption in Large-Scale Sensor Networks through Distributed Data Compression and Hierarchical Aggregation," *IEEE J. Selected Areas in Comm.,* vol. 22, no. 6, pp. 1130-1140, Aug. 2004.

[5] C.M. Sadler and M. Martonosi, "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems,* Nov. 2006.

[6] Y. Xu and W.-C. Lee, "Compressing Moving Object Trajectory in Wireless Sensor Networks," *Int'l J. Distributed Sensor Networks,* vol. 3, no. 2, pp. 151-174, Apr. 2007.

[7] 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.

[8] 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,* vol. 47, no. 2, pp. 146-153, June 2009.

[9] 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.

[10] 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.

[11] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory Pattern Mining," *Proc. ACM SIGKDD,* pp. 330-339, 2007.

[12] 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.

[13] L. Chen, M. Tamer Özsu, and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories," *Proc. ACM SIGMOD,* pp. 491-502, 2005.

[14] 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.

[15] 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.

[16] H.-P. Tsai, D.-N. Yang, W.-C. Peng, and M.-S. Chen, "Exploring Group Moving Pattern for an Energy-Constrained Object Tracking Sensor Network," *Proc. 11th Pacific-Asia Conf. Knowledge Discovery and Data Mining,* May 2007.

[17] C.E. Shannon, "A Mathematical Theory of Communication," *J. Bell System Technical,* vol. 27, pp. 379-423, 623-656, 1948.

[18] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.,* pp. 3-14, 1995.

[19] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu, "Freespan: Frequent Pattern-Projected Sequential Pattern Mining," *Proc. ACM SIGKDD,* pp. 355-359, 2000.

[20] 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.

[21] 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.

[22] 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.

[23] V. Guralnik and G. Karypis, "A Scalable Algorithm for Clustering Sequential Data," *Proc. First IEEE Int'l Conf. Data Mining,* pp. 179-186, 2001.

[24] J. Yang and W. Wang, "CLUSEQ: Efficient and Effective Sequence Clustering," *Proc. 19th Int'l Conf. Data Eng.,* pp. 101-112, Mar. 2003.

[25] 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.

[26] 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.

[27] 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.

[28] "Stargate: A Platform x Project," http://platformx.sourceforge.net, 2010.

[29] "Mica2 Sensor Board," http://www.xbow.com, 2010.

[30] 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.

[31] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer,* vol. 34, no. 8, pp. 57-66, Aug. 2001.

[32] 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.

[33] D. Ron, Y. Singer, and N. Tishby, "Learning Probabilistic Automata with Variable Memory Length," *Proc. Seventh Ann. Conf. Computational Learning Theory,* July 1994.

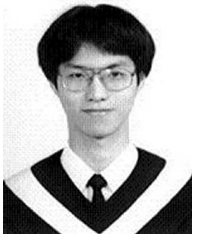
[34] D. Katsaros and Y. Manolopoulos, "A Suffix Tree Based Prediction Scheme for Pervasive Computing Environments," *Proc. Panhellenic Conf. Informatics,* pp. 267-277, Nov. 2005.

[35] 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.

[36] 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-V562, 2004.

[37] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," *Information Processing Letters,* vol. 76, nos. 4-6, pp. 175-181, 2000.
[38] A. Strehl and J. Ghosh, "Cluster Ensembles—A Knowledge Reuse Framework for Combining Partitionings," *Proc. Conf. Artificial Intelligence,* pp. 93-98, July 2002.
[39] 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.
[40] G. Saporta and G. Youness, "Comparing Two Partitions: Some Proposals and Experiments," *Proc. Computational Statistics,* Aug. 2002.
[41] 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.
[42] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," *Computer,* special issue on sensor networks, vol. 37, no. 8, pp. 41-49, Aug. 2004.
[43] 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.
[44] 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.
[45] 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.
[46] 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.
[47] 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.
[48] 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.
[49] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Ultra-Low Power Data Storage for Sensor Networks," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks,* pp. 374-381, Apr. 2006.
[50] P. Dutta, D. Culler, and S. Shenker, "Procrastination Might Lead to a Longer and More Useful Life," *Proc. Sixth Workshop Hot Topics in Networks,* Nov. 2007.
[51] Y. Diao, D. Ganesan, G. Mathur, and P.J. Shenoy, "Rethinking Data Management for Storage-Centric Sensor Networks," *Proc. Third Biennial Conf. Innovative Data Systems Research,* pp. 22-31, Nov. 2007.
[52] F. Osterlind and A. Dunkels, "Approaching the Maximum 802.15.4 Multi-Hop Throughput," *Proc. Fifth Workshop Embedded Networked Sensors,* 2008.
[53] S. Watanabe, "Pattern Recognition as a Quest for Minimum Entropy," *Pattern Recognition,* vol. 13, no. 5, pp. 381-387, 1981.
[54] L. Yuan and H.K. Kesavan, "Minimum Entropy and Information Measurement," *IEEE Trans. System, Man, and Cybernetics,* vol. 28, no. 3, pp. 488-491, Aug. 1998.
[55] 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.
[56] D. Bolier, "SIM : A C++ Library for Discrete Event Simulation," http://www.cs.vu.nl/eliens/sim, Oct. 1995.
[57] 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.
[58] 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.
[59] 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.
[60] C. Largeron-Leténo, "Prediction Suffix Trees for Supervised Classification of Sequences," *Pattern Recognition Letters,* vol. 24, no. 16, pp. 3153-3164, 2003.

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 (EE), 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 worked at the Internet Research Lab with advisor Prof. Wanjiun Liao. Afterward, he joined the Network Database Lab with advisor Prof. Ming-Syan Chen as a post-doctoral 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 Research Center of Information Technology Innovation (CITI) in the Academia Sinica, Taiwan, and is also a distinguished professor jointly appointed by the EE Department, CSIE Department, and Graduate Institute of Communication Engineering (GICE) at 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 in 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. He has published more than 300 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* and also the *Journal of Information Systems Education*, is currently the editor in chief of the *International Journal of Electrical Engineering* (IJEE), and is on the editorial board of the *Very Large Data Base (VLDB) Journal* and the *Knowledge and Information Systems (KAIS) Journal*. He holds, or has applied for, 18 US patents and seven ROC patents in his research areas. He is a recipient of the National Science Council (NSC) Distinguished Research Award, Pan Wen Yuan Distinguished Research Award, Teco Award, Honorary Medal of Information, and 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.