# Data Compression on Group Movement Patterns

[1]P Mahesh Kumar, [2]M Jayapal, [3]V Shiva Narayana Reddy, [4]K Karthik, [5]J Swathi
[1]Dept. of CSE, BSIT, Chevella, Hyderabad, AP, India
[3]VBITS, Nadergul, Hyderabad, AP, India
[4]Dept. of CSE & IT, TCET, Peddapally, Karimnagar, AP, India
[5]Dept. of CS, VITS, Karimnagar, AP, India

**Abstract**
In this work, we exploit the characteristics of group movements to discover the information about groups of moving objects in tracking applications. 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. With the discovered information, we propose a compression algorithm, called 2P2D algorithm, 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. 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. We are using these algorithms to find out the performances on the bases on time complexity and space complexity by using visualization technique.

**Keywords**
Data Mining, Data Compression, Distributed Clustering, Object Tracking

## I. 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 [1-6]. As a result we have millions of databases today used in business management, Government administration, Scientific & engineering data management, marketing management, fraud detection and in many other applications. Today the number of such databases has increased and beeps growing rapidly because of powerful relational database systems. This exclusive growth in databases in modern world raises the need of data analysis. The business organizations, Government Organizations & Scientific organizations by using relational database system maintains large repository of volumes of data related to the business transactions. In recent years people want to study the existing large volumes of data, comparing the data and using this comparison they are taking decisions for improving the business growth of the organizations. This is called Data analysis. This Data analysis is becoming more and more important to implement decision support systems. To perform this data analysis efficiently and effectively today we need new techniques and tools that can intelligently and automatically transform the processed data into useful information and knowledge. This process is called data mining. As a result data mining has become a research area with increasing importance. Data mining also referred to as knowledge discovery in database is a process of extraction of unknown and potentially useful information from the databases. This information is called knowledge. Data mining is a process of extraction hidden patterns from the existing data. It is also called knowledge mining from databases, knowledge extraction, data archeology, and data dredging. Data mining is the process of discovering meaningful new correlation patterns and trends by analyzing large amounts of data stored in repositories using pattern recognition technologies, statistical & mathematical techniques. The combination of data mining with the help of heuristic searches, artificial intelligence and expert systems is more efficient and effective to search the data from large volumes of repository. By knowledge discovery in databases, intensity knowledge, resultant or any high level information from large volumes of data in databases and this knowledge information can be investigated from different angles, in different views, with different parameters.

Mining information knowledge from large databases has been recognized in the industry area as a key research topic, the discovered knowledge can be applied to information management, query processing, decision making, and process control and in many other applications. Data mining tools predict future tense and behavior. This allows business organizations to make proactive and knowledge driven decisions. Data Mining tools can answer business questions the traditionally time consuming to resolve the search databases for hidden pattern finding predictive information that experts required to take decisions.

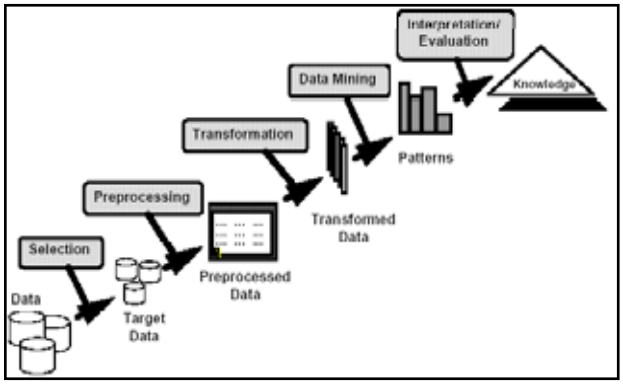


Fig. 1: Data Mining as a Step in an Interactive Knowledge Discovery Process

Researchers in many different fields including database systems, knowledge base systems, artificial intelligence, machine learning, knowledge acquisition, statistics, special databases and data visualization have shown great interest in data mining. Furthermore several applications in information providing services such as online services and World Wide Web also require various data mining techniques to better understand user behavior, to improve the service provided and to improve the business services. In

response to such a demand and need our system (paper) is to provide an implementation of clustering techniques to classify the data efficiently and effectively from large volumes of data.

## II. Related Work

### A. Movement Pattern Mining

Agrawal and Srikant [7] 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 [8], 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 [10]. Discover path traversal patterns in a Web environment, while Peng and Chen [11] mine user moving patterns incrementally in a mobile computing system. However, sequential patterns and its variations like [10-11] 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 [12-13]. 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.

### B. 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 [16], but the above dynamic programming approaches suffer from scalability problem [17]. To provide scalability, approximation or summarization techniques are used to represent original data. Guralnik and Karypis [17] 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 [18], which iteratively identifies a sequence to a learned model, yet the generated clusters may overlap which differentiates their problem from ours.

### C. 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.

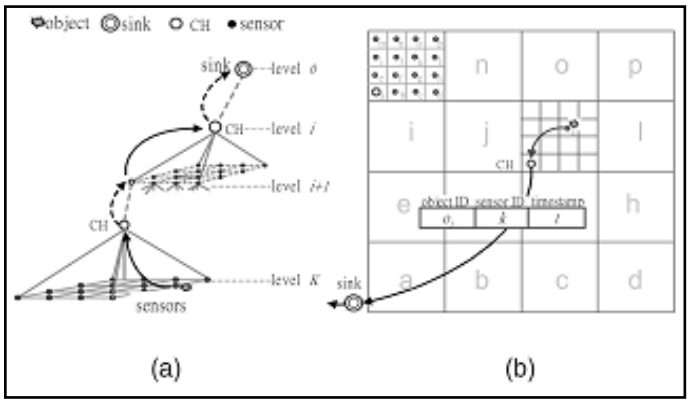


Fig. 2: (a) The hierarchical- and cluster-based network structure and the data flow of an update-based tracking network. (b) A flat view of a two layer network structure with 16 clusters

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.

## III. 2P2D Compression Algorithm

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 [19-20]. 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 [21-25], 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. 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 compress the location sequences of moving objects elaborately. 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 post processing. 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 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.

### A. 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.

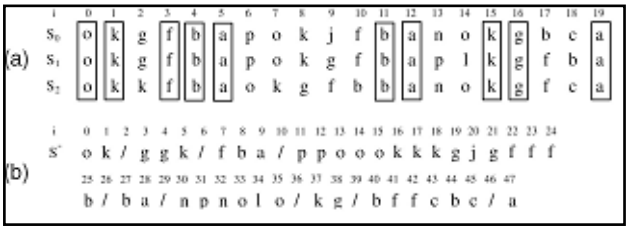


Fig. 3: An example of the Merge algorithm. (a) Three sequences with high similarity. (b) The merged sequence S``

Consider an illustrative example in fig. 3(a), where three location sequences S0, S1, and S2 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 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. 3b 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.

Table 1: Description of the Notations

| Notation | Description |
|---|---|
| $\Sigma$ | The alphabet of sensor IDs. |
| $\sigma$ | A symbol in $\Sigma$. |
| $S$ | 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 $\sigma$ in $S$. |
| $n_{hit}(\sigma)$ | The number of predictable items of $\sigma$ in $S$. |
| $\hat{s}$ | The sub-set of $\Sigma$, where $n_{hit}(\sigma) > 0$ for every $\sigma \in \hat{s}$. |
| $p_i$ | The occurrence probability of $\sigma_i$ corresponding to $S$. |

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 S2 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 include 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:

```
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 =S0[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. 4: The Merge Algorithm

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, $|Sj[i] - \sigma| \leq eb$ must hold for $0 \leq j < n$, where n is the number of sequences.

Therefore, to compress the location sequences for a group of moving objects, we propose the Merge algorithm shown in Fig. 4. The input of the algorithm contains a group of sequences {Si|0 ≤ 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 column wise 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. 3a, 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.

### B. 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 [26], 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.

```
Algorithm: Replace
Input: a sequence: S
       a predictor: Tg
Output: an intermediate sequence: S'
0.   ŝ = ϕ
1.   n(".") = 0
2.   taglst = get_taglst(S, Tg)
3.   for 0 ≤ i < |S|-1                    /* getting statistics of S */
4.       σ = S[i]
5.       n(σ)++
6.       if taglst[i] == 1 then
7.           nhit(σ) = nhit(σ) + 1
8.           if σ ∉ ŝ then
9.               add σ to ŝ
10.  for ∀σ ∈ ŝ                          /* the accumulation rule */
11.      if n(σ) == nhit(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 nhit(σ) > n(σ) - n(".") then
19.                  replaceHitItems(S, taglst, σ)
20.                  n(".") = n(".") + nhit(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(".") + nhit(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. 5: The Replace Algorithm

Fig. 5, shows the Replace algorithm. The input includes a location sequence S and a predictor Tg, 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 rules. In our design, since finding a combination of predictable symbols to make gain(ŝ`) > 0 hold is more costly, the algorithm is prone to replace symbols with the 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 ^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 ŝ` makes gain(ŝ`) > 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 ŝ` contains all of the symbols in ŝ.

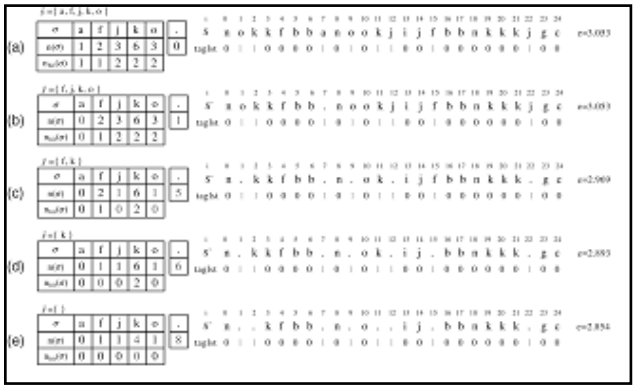

Fig. 6: An Example of the Replace Algorithm

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. 6, our algorithm generates the statistic table and taglst shown in Fig. 6a. In this example, ŝ = {k, j, o, a, f}, and the numbers of items of the symbols are 6, 3, 3, 1, 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 nhit(`a`) (Lines 10-14). After that, the statistic table is updated, as shown in Fig. 6b. Second, according to the multiple symbol rule, we replace the predictable items of 0j0 and 0o0 simultaneously such that the entropy of S` is reduced to 2.969. Next, because n(`f`) is less than n(`a`), 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 nhit(`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 S0 with entropy 2.854. In this example, all predictable items are replaced to minimize the entropy.

In addition, for the example shown in fig. 3, the Replace algorithm reduces S0, S1, and S2's entropies from 3.171, 2.933, and 2.871 to 2.458, 2.828, and 2.664, respectively, and encoding S0`, S1`, and

S2` reduces the sum of output bit streams 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.

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 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. 7a, the segments E1, E2, and E3 are aligned and named G-segments, whereas segments A, B, C, and D are named S-segments. Fig.s 7b, 7c, and 7d show an illustrative example to construct the frame for the three sequences. First, the Merge algorithm combines E1, E2, and E3 to generate an intermediate sequence śE. Next, śE together with A, B, C, and D is viewed as a sequence and processed by the Replace algorithm to generate an intermediate sequence ś, which comprises śA, śB, śC, śD, and śE. Finally, intermediate sequence ś is compressed and packed.

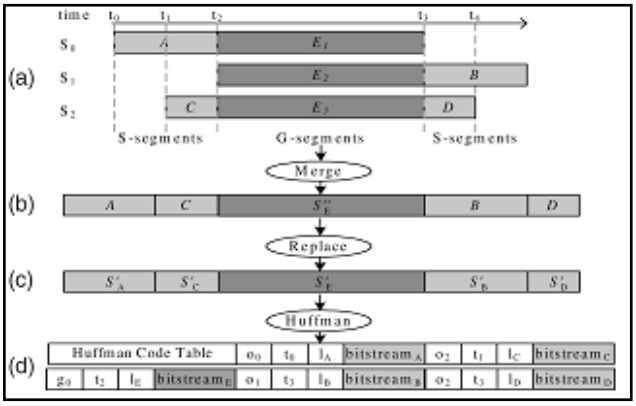


Fig. 7: An example of constructing an update packet. (a)Three sequences aligned in time domain. (G-segments: E1, E2, and E3, S-segments: A, B , C, and D.) (b) Combining G-segments by the Merge algorithm. (c) Replacing the predictable items by the Replace algorithm. (d)Compressing and packing to generate the payload of the update packet

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 × 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 + Di × b + c + l) where Di 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 × D × b +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 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 × ( H + a + b + c ) × (1- p ) × 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.

## IV. Conclusion

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 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. 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.

## References

[1] S.S. Pradhan, J. Kusuma, 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 , 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, 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, 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, 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] R. Agrawal, R. Srikant,"Mining Sequential Patterns", Proc. 11th Int'l Conf. Data Eng., pp. 3-14, 1995.
[8] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. Hsu,"Freespan: Frequent Pattern-Projected Sequential Pattern Mining," Proc. ACM SIGKDD, pp. 355-359, 2000.
[9] J. Yang, 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.-S. Chen, J.S. Park, P.S. Yu,"Efficient Data Mining for Path Traversal Patterns," Knowledge and Data Eng., Vol. 10, No. 2, pp. 209-221, 1998.
[11] W.-C. Peng, 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.
[12] 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.
[13] 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.
[14] Y. Wang, E.-P. Lim, 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, D. Pedreschi,"Time-Focused Clustering of Trajectories of Moving Objects", J. Intelligent Information Systems, Vol. 27, No. 3, pp. 267-289, 2006.
[16] L. Chen, M. Tamer O¨ zsu, V. Oria,"Robust and Fast Similarity Search for Moving Object Trajectories", Proc. ACM SIGMOD, pp. 491-502, 2005.
[17] V. Guralnik, G. Karypis,"A Scalable Algorithm for Clustering Sequential Data", Proc. First IEEE Int'l Conf. Data Mining, pp. 179- 186, 2001.
[18] J. Yang, W. Wang,"CLUSEQ: Efficient and Effective Sequence Clustering", Proc. 19th Int'l Conf. Data Eng., pp. 101-112, Mar. 2003.
[19] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless Sensor Networks: A Survey", Computer Networks, Vol. 38, No. 4, pp. 393-422, 2002.
[20] D. Culler, D. Estrin, M. Srivastava,"Overview of Sensor Networks," Computer, special issue on sensor networks, Vol. 37, No. 8, pp. 41-49, Aug. 2004.
[21] H.T. Kung, D. Vlah,"Efficient Location Tracking Using Sensor Networks", Proc. Conf. IEEE Wireless Comm. and Networking, Vol. 3, pp. 1954-1961, Mar. 2003.
[22] 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.
[23] W. Zhang, 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.
[24] J. Yick, B. Mukherjee, D. Ghosal,"Analysis of a Prediction-Based Mobility Adaptive Tracking Algorithm", Proc. Second Int'l Conf. Broadband Networks, pp. 753-760, Oct. 2005.
[25] C.-Y. Lin, W.-C. Peng, 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.
[26] C.E. Shannon,"A Mathematical Theory of Communication", J. Bell System Technical, Vol. 27, pp. 379-423, 623-656, 1948.