Available online at www.sciencedirect.com

ScienceDirect

The Journal of Systems and Software 80 (2007) 1678–1698

The Journal of Systems and Software

www.elsevier.com/locate/jss

Energy efficient strategies for object tracking in sensor networks: A data mining approach ☆

Vincent S. Tseng *, Kawuu W. Lin

Institute of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, ROC

Received 8 March 2006; received in revised form 23 October 2006; accepted 14 December 2006
Available online 13 January 2007

Abstract

In recent years, a number of studies have been done on object tracking sensor networks (OTSNs) due to the wide applications. One important research issue in OTSNs is the energy saving strategy in considering the limited power of sensor nodes. The past studies on energy saving in OTSNs considered the object's movement behavior as randomness. In some real applications, however, the object movement behavior is often based on certain underlying events instead of randomness completely. In this paper, we propose a novel data mining algorithm named *TMP-Mine* with a special data structure named TMP-Tree for efficiently discovering the *temporal movement patterns* of objects in sensor networks. To our best knowledge, this is the first work on mining the movement patterns associated with time intervals in OTSNs. Moreover, we propose novel location prediction strategies that utilize the discovered temporal movement patterns so as to reduce the prediction errors for energy savings. Through empirical evaluation on various simulation conditions and real dataset, TMP-Mine and the proposed prediction strategies are shown to deliver excellent performance in terms of scalability, accuracy and energy efficiency.

Keywords: Location prediction; Temporal movement patterns; Object tracking; Sensor networks; Data mining

1. Introduction

Energy efficient tracking of objects in sensor networks is an emerging research field attracting a lot of attention recently. Advances in wireless communication and microelectronic device technologies have enabled the development of low-power micro-sensors and the deployment of large scale sensor networks. With the capabilities of pervasive surveillance, sensor networks are applied in a lot of commercial and military applications, like the object tracking application and the environmental data collection. However, the intrinsic limitations such as power constraints, synchronization, deployment, and data routing bring numerous research challenges (Akyildiz et al., 2002; WINS project).

In a sensor network, the deployed sensor nodes form *ad hoc* networks (Akyildiz et al., 2002; Hara et al., 2004) and the nodes can communicate with each other by RF radios without special infrastructure. Compared with the standard ad hoc networks, a sensor network has the following characteristics: (1) the sensor nodes are static in terms of physical location; (2) the computing power is normally weak; (3) the energy carried in a sensor node is limited. Due to the environmental conditions that replenishing the battery charge is expensive or infeasible, the energy is one of the most important system resources that should be reserved (Carle and Simplot, 2004). In this paper, we focus on the problem of energy saving in the object tracking sensor networks (OTSNs).

☆ This paper is an extended version of Tseng and Lin (2005), entitled "Mining Temporal Moving Patterns in Object Tracking Sensor Networks", by V. S. Tseng and K. W. Lin, which appeared in Proceedings of the International Workshop on Ubiquitous Data Management (held with ICDE'05), April, 2005, Tokyo, Japan.

* Corresponding author. Tel.: +886 6 2757575x62536; fax: +886 62747076.
E-mail address: tsengsm@mail.ncku.edu.tw (V.S. Tseng).

In an OTSN, each sensor node is composed of sensing, data processing, and communication components (Raghunathan et al., 2002). Nevertheless, the power required by different sensing components varies widely. For example, using a velocity-based strategy to track the moving objects requires the velocity sensing component, which is an energy expensive device and is not the necessary equipment for all sensor nodes. Hence, one of our research goals is to propose energy efficient strategies by using intelligent software mechanism instead of adding the energy expensive components.

A number of past studies tried to solve the energy saving issue from the *hardware design*. For instance, the optimization problem of the communication cost by inactivating the RF radios of idle sensor nodes was widely discussed (Goel and Imielinski, 2001; Heinzelman et al., 2000). However, these studies did not consider the energy saving issues for these components (Xu et al., 2004) although the sensing and computing components consume relative less energy than radios (Raghunathan et al., 2002). Several researchers tried to save the energy through the software approach like scheduling of sensors. One of the novel ideas is to put a sensor node into sleeping mode when there are no objects in its coverage/sensing region, and a sensor node is activated again whenever an object enters its sensing region. Based on this idea, the studies for energy saving in OTSNs can be further divided into two categories: *non-prediction based tracking* and *prediction based tracking*. The intuitive way of non-prediction based tracking method is periodically turn the sensor nodes off and only activate the sensor nodes when it is time to monitor their sensing regions. Another non-prediction based tracking method is planting an agent onto the mobile device, named mobile agent. With the help of mobile agents, the communication and sensing overheads can be greatly reduced (Tseng et al., 2004). The prediction based methods use the information of a moving object like velocity or moving direction to predict the next location the object might visit.

Note that both of the non-prediction based and prediction based tracking methods neglected the event characteristic of objects. In some real applications, the behavior of the moving objects is often based on certain underlying events instead of randomness completely. For example, consider the bus tracking project in Mani (2003), the route of each bus is pre-specified rather than being random. Central to this issue is the problem of discovering the movement behavior of objects. The wireless technologies nowadays have allowed the collection of large amount of movement logs (CRAWDAD; Reality Mining Project). Therefore, it is feasible to discover the hidden knowledge like movement behavior from the wireless log. Over the past few years a considerable number of studies have been done on using data mining techniques to discover this kind of interesting patterns/rules from World Wide Web (Pei et al., 2000), transactional databases (Agrawal and Srikant, 1995) and mobility databases (Huang et al., 2003; Kyriakakos et al., 2003; Tseng and Lin, 2006; Tseng and Tsui, 2004). Note that the discovered patterns in such applications are implicitly assumed to be valid for some period until the mobility patterns change with time. To keep the patterns being updated, the data mining techniques may be applied on the most updated log periodically. Most of these past studies focused only on the aspect of path analysis and only few of them (Wu et al., 2001) considered the temporal characteristic that is very critical in wireless networks. Without considering the temporal information, the important knowledge may be overlooked (Roddick and Spiliopoulou, 2002).

Take a vehicle tracking application as example. Assume that each car is attached with a receiver that can receive the beacon of the sensor node the car visits. By collecting the log of cars, we may use the data mining method to discover temporal movement rules. Suppose the following rules are discovered: Rule_1: (Station $A \rightarrow$ interval 10 min $\rightarrow$ Station $B \rightarrow$ interval 5 min $\rightarrow$ Station C); Rule_2: (Station $A \rightarrow$ interval 20 min $\rightarrow$ Station $B \rightarrow$ interval 5 min $\rightarrow$ Station D). By dispatching these rules to the corresponding sensor nodes, the tracking can be mode in more energy efficient way. For instance, if a car moves with the pattern as (Station $A \rightarrow$ interval 10 min $\rightarrow$ Station $B \rightarrow$ interval 5 min) that matches with Rule_1, the node in Station B has only to activate the node in Station C rather than that in Station D or those around Station B. As can be seen, the temporal clues can effectively enhance the prediction accuracy in an OTSN. By integrating the *temporal movement patterns* (*TMPs*) into the prediction strategies, the number of sensor nodes that are incorrectly and unnecessarily activated is expected to be substantially reduced and more energy can be saved in an OTSN.

However, no studies have explored the issue of discovering objects' temporal movement patterns in OTSNs so as to enhance the energy efficiency. In this paper, we propose a novel data mining method named *TMP-Mine* with a special data structure named *TMP-Tree* for efficiently discovering TMPs in OTSNs. To our best knowledge, this is the first work on mining the movement patterns with time intervals in OTSNs. Moreover, we propose two prediction strategies for predicting the location of a missing object in OTSNs by utilizing TMPs. The first prediction strategy named *PTMP* is capable of making prediction by employing *TMP*s with no need to detect the object velocity. Hence, it can be applied to the sensor networks with low-end sensor nodes. The second strategy, namely $PES + PTMP$, is a hybrid approach by integrating *PTMP* method with a popular velocity-based strategy named *PES* (Xu et al., 2004). This integrated strategy can further enhance the energy efficiency if the sensor nodes carry the velocity detection capability. Through empirical evaluation on various simulation conditions and real dataset, TMP-Mine and the proposed prediction strategies are shown to deliver excellent performance in terms of scalability, accuracy and energy efficiency.

The rest of this paper is structured as follows. We briefly review the related work in Section 2. In Section 3, we describe the overall system architecture and workflow. In

Section 4, we describe the problem definition and the proposed data mining algorithm, namely TMP-Mine, is presented in Section 5. Section 6 gives the detailed description on the prediction strategies. The empirical evaluation for performance study is made in Section 7. The conclusions and future directions are given in Section 8.

2. Related work

In this section, we review the past studies on the three subjects closely related to this research, namely energy efficient strategies for object tracking, behavior mining and behavior prediction.

For energy saving policies in sensor networks, a number of past studies tried to solve this issue from the aspect of *hardware design*. For instance, the optimization problem of the communication cost by inactivating the RF radios of idle sensor nodes was widely discussed (Goel and Imielinski, 2001; Heinzelman et al., 2000). There are also a lot of research efforts in energy efficient media access control (MAC) (Shih et al., 2001; Woo and Culler, 2001; Ye et al., 2002). Several researches tried to save the energy through the software design approach. In Cerpa et al. (2001), the authors proposed the *Frisbee* scheme that keeps only a limited zone of the network called a Frisbee that is close to the moving object in the fully active state. However, it is difficult to choose a good radius of the Frisbee. In Lin et al. (2006), the authors developed some tree structures for efficient object tracking by considering the physical network structure. In Xu et al. (2004), Xu *et al.* proposed a network model, in which a sensor node is activated only if there exist some objects in its coverage region. Besides, the activated node is scheduled to be active for X seconds and in sleeping mode for $(T - X)$ seconds during the T seconds periodically to save the energy. They also proposed a set of prediction based energy saving schemes named *PES* to conserve the scarce energy resource by using the latest detected or average velocity of a missing object to predict its current location. To select the object velocity and direction, three models named *Heuristics INSTANT*, *Heuristics AVERAGE*, and *Heuristics EXP_AVG* were also proposed. In the prediction phase, three mechanisms were proposed, namely *Heuristics DESTINATION*, *Heuristics ROUTE*, and *Heuristics ALL_NBR*. The Heuristics DESTINATION utilizes only the velocity information for activation while the Heuristics ROUTE activates all nodes on the route. The Heuristics ALL_NBR mechanism activates all neighboring nodes of the destination. However, the latest detected velocity of objects may be incorrect since the sensor node might lose the object in its periodical sleeping mode. Hence, the PES method still incurs the problem of incorrect prediction.

For the research on behavior mining, numerous studies have been done on mining users' behavior patterns like association rules or sequential patterns in WWW (Pei et al., 2000) and transactional databases (Agrawal and Srikant, 1995; Heinzelman et al., 2000). In Pei et al. (2000), the authors proposed a method named WAP-Mine for fast discovery of the web access patterns from web logs by using a tree-based data structure without candidate generation. Previous studies on the mining of temporal databases include (Agrawal and Srikant, 1995; Ale and Rossi, 2000; Guil et al., 2004; Li et al., 2003; Padmanabhan and Tuzhilin, 1996; Srikant and Agrawal, 1996). In Agrawal and Srikant (1995), the authors proposed a method for mining the transactions to discover the time-ordered patterns named sequential patterns. In Srikant and Agrawal (1996), the method using sliding window to restrict the time gap between sets of transactions in mining sequence patterns was proposed. The issue of using the temporal logic and related operators such as *since*, *until* and *next* was explored in Padmanabhan and Tuzhilin (1996). In the category of mobility mining, most of the existing researches focused only on the analysis of user movement behavior Lee and Wang (2003), Yavas et al. (2005). To discover the patterns from two-dimensional mobility data, the problem of mining location associated service patterns was first studied by Tseng and Tsui (2004). A novel method for discovering users' sequential movement patterns associated with requested services in mobile web systems was also proposed by Tseng and Lin (2006).

In the area of behavior prediction, some researchers proposed variations of Markov models, such as Dependency Graph (DG) Padmanabhan and Mogul (1996), Prediction-by-Partial-Match (PPM) Palpanas and Mendelzon (1999) and N-gram model Su et al. (2000), for predicting the user behavior in WWW. Basically, these methods employ the last N page views of the user to predict the next view by using the patterns discovered. Yang et al. (2004) studied the association-rule based sequential classifiers and considered features of association rules such as order, adjacency, and recency systematically to construct prediction models from web logs.

3. Problem statement

In this section, we first state the problem. Afterwards, we describe the network environments and the behavior issues of moving objects. The performance metrics are also described in the end of this section.

In this work, we adopt a network model for OTSNs as proposed in Xu et al. (2004), in which a sensor node is activated only if there is object in its coverage/sensing region. Besides, the activated node is scheduled to be in active mode for X seconds and in sleeping mode for $(T - X)$ seconds during the T seconds periodically to save the energy. Moreover, we assume the movement log of objects is collectable (Mani, 2003; Tseng and Lin, 2005) and the trajectory of each object is represented in the form of $S = \langle (l_1, t_1)(l_2, t_2), \ldots, (l_n, t_n) \rangle$, where l_i represents the sensor node location at time t_i. The log is considered as a valuable resource since it contains the habitual patterns of objects. The targeted problem is to two fold: (1) efficient discovery of temporal movement patterns (TMPs) for objects, and (2) location prediction by utilizing TMPs for energy saving.

To solve the problem described above, we shall discover TMP in the form as $P = \langle(l_1, i_1, l_2, i_2, \ldots, i_{r-1}, l_r)\rangle$ where i_k semantically represents the time interval between two traversed locations. Moreover, we shall generate temporal movement rule (TMR) in the form of

$$R_t = \langle(l_1, i_1, l_2, i_2, \ldots, l_{m-1}, i_{m-1})\rangle \rightarrow \langle(l_m)\rangle$$

for incorporation into the location prediction mechanisms so as to achieve low energy and low missing rate in the OTSNs.

Note that we assume the behavior of moving objects is often based on certain underlying events instead of randomness completely (Tseng and Lin, 2005, 2006; Tseng and Tsui, 2004; Yavas et al., 2005). An event is a stream of locations with time intervals. Note that the characteristics of events in OTSNs include not only location but also time interval. In our network model, the movement behavior of objects may be decided by certain events or be random. Detailed network model will be given in Section 7.1.

In solving the targeted problem, some important performance metrics should be considered. In this work, we adopt two popular metrics named *Total Energy Consumed* (*TEC*) (Xu et al., 2004) and *missing rate* (Xu et al., 2004). *TEC* indicates the total energy consumed by sensor nodes in the OTSN during data mining phase and object tracking phase. *Missing rate* denotes the number of erroneous predictions in a specified time period in ratio of the total number of movement of objects. Obviously, low *TEC* and low missing rate can benefit the lifetime of the whole network, and this is the aimed goal for this research.

4. System architecture

Fig. 1 shows the proposed system architecture. We assume that the movement of objects in wireless sensor networks is recorded in the system logs (Mani, 2003; Tseng and Lin, 2005). In our proposed network model, each object is able to record the sensor nodes it visited together with the arrival time at each node. To collect the movement log, several powerful sensor nodes equipped with storage devices are deployed over the *outer* of the network for retrieving the log of each object that is exiting from the network. The system workflow consists of two main phases: (1) data mining phase, and (2) object tracking phase. At first, the sensor network collects and integrates the movement log of moving objects. Then the integrated movement log is used as the input to the data mining method named TMP-Mine for discovering the TMPs. By performing the proposed TMP-Mine algorithm, the TMPs will be discovered and then the TMRs are generated for use by location prediction so as to track objects in energy efficient manner. The two phases are described in details in below:

1. Data mining mechanism: The data mining mechanism consists of three components, namely data integrator, TMP-Miner and TMR generator. Because the logs are distributed in the surrounding sensors of the network, a data integrator is required to integrate and preprocess the logs collected before the data mining algorithm is applied on the logs. Table 1 is an example showing the prepared movement log with time intervals between visiting to sensor nodes. Take the fourth tuple, $\{(f, 0)(e, 5)(b, 13)\}$, as example, it means the object arrived in the region of sensor node *f*, *e* and *b* at time point 0, 5 and 13, respectively.Once the log is prepared, TMP-Mine algorithm will be applied to discover the TMPs from the integrated log. The functionality of rule generator is to generate the TMRs from discovered TMPs according to some parameters like confidence. Afterwards, the TMRs are utilized by the location prediction strategies so as to achieve the goal for energy savings. Moreover, the rule generator will also evaluate the strength of each TMR for rule ranking (described in details in Section 5.5).
2. Object tracking mechanism: The spirit of the proposed tracking mechanism is to predict the next location of each object by utilizing the TMRs. Before activating

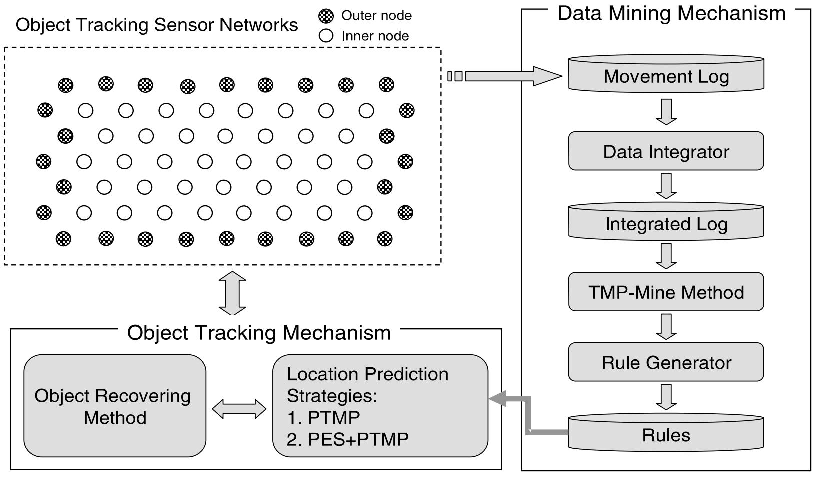


Fig. 1. System architecture.

Table 1
An integrated log of temporal moving sequences

Object ID	Temporal moving sequence
1	$\langle(a,1)(e,3)(c,5)(b,10)\rangle$
2	$\langle(a,3)(b,5)(c,7)(d,12)\rangle$
3	$\langle(a,1)(e,2)(c,5)(b,10)\rangle$
4	$\langle(f,0)(e,5)(b,13)\rangle$
5	$\langle(a,4)(b,6)(c,7)(d,12)\rangle$
6	$\langle(f,0)(a,4)(c,6)(d,10)\rangle$
7	$\langle(a,0)(b,1)(c,2)(d,6)\rangle$
8	$\langle(f,1)(e,3)(b,8)(d,14)\rangle$

the object tracking mechanism, we should dispatch the TMRs to appropriate sensor nodes. In considering the property that the storage associated with each sensor node is limited, we dispatch the TMRs to the sensor nodes according to the *location-based criterion*. That is, only the TMRs that are location related to a sensor node will be loaded into that node. Take the TMR, $(f,5)\to(b)$, for example, we would deploy this TMR at only node f and its neighboring sensors rather than all the sensors in the network. Dispatching TMRs by the location-based criterion as described above will greatly decrease the demands of storage for the sensor nodes.The tracking mechanism is composed of the location prediction strategy and object recovering method. For location prediction strategies, we propose two approaches named PTMP and PES + PTMP. *PTMP* performs location prediction by employing TMRs with no need to detect the object velocity. Hence, it can be applied to the sensor networks with low-end sensor nodes. The approach *PES + PTMP* is a hybrid one by integrating *PTMP* method with a popular velocity-based strategy named *PES* (Xu et al., 2004). Recall that a sensor node is periodically activated when an object is in its coverage region according to the scheduling policy. Under such environments, the prediction mechanism will be triggered whenever a sensor node loses an object. If the prediction mechanism fails to recover the object within a specified deadline, the *flooding* (Cerpa et al., 2001) strategy will be activated for recovering the missing object.

5. Proposed data mining algorithm: TMP-mine

In this section, we first formulate the mining problem and then propose a novel algorithm named TMP-Mine that can discover the TMPs efficiently. How the TMRs are generated is also described. We illustrate the discovery of TMPs by an elaborate example in the end of this section.

5.1. Formulation of mining problem

Let $S=\langle(l_1,rt_1)(l_2,rt_2)\ \ldots\ (l_m,rt_m)\rangle$ be a temporal movement sequence of an object with length equal to m, where l_i represents the object's location at time rt_i and $rt_i< rt_{i+1}\forall 1\leqslant i<m$. The ascending order of elements in a sequence is decided by using the time as the key. In order to discover the *temporal* movement patterns, we use the *time slot* to uniformly segment the time dimension of a sequence. If the time slot is set to b, we will obtain a transformed sequence $S_t=\langle(l_1,t_1)(l_2,t_2)\ldots,(l_m,t_m)\rangle$, where $t_i=\lfloor\frac{rt_i}{b}\rfloor$.

Definition 1. A temporal movement sequence $S'=\langle(l'_1,t'_1)(l'_2,t'_2),\ldots,(l'_m,t'_m)\rangle$ is a *sub-pattern* of another access pattern $S=\langle(l_1,t_1)(l_2,t_2),\ldots,(l_n,t_n)\rangle$, written as $S'\subset S$, if $m\leqslant n$ and there exists a strictly increasing sequence $(k_1,k_2,\ldots,k_m)$ of indices such that for all $j=1,2,\ldots,m$, $l'_j=l_{k_j}$ and $t'_{j+1}-t'_j=t_{k_{j+1}}-t_{k_j}$. Here, S is called the *super-pattern* of S'.

Definition 2. Given a database $D=\{S_1,S_2,\ldots,S_N\}$ that contains N temporal movement sequences, the *support* of sequence S is defined as

$$\sup(S)=\frac{|\{S_i|S\subset S_i \text{ and } 1\leqslant i\leqslant N\}|}{N}.$$

Definition 3. $S=\langle(l_1,t_1)(l_2,t_2),\ldots,(l_r,t_r)\rangle$ is called a *frequent temporal movement sequence* if $\sup(S)$ is greater than or equal to a specified support threshold δ, and the corresponding TMP is written as $P=\langle(l_1,i_1,l_2,i_2,\ldots,i_r,l_r)\rangle$, where i_k semantically represents the time interval between l_k and l_{k+1} visiting and $i_j=t_{j+1}-t_j$.

With the above definitions, the problem of discovering TMPs is defined as follows. Given a database D containing the temporal movement sequences of objects and a specified support threshold δ, the problem is to discover all the TMPs existing in this database. In this research, we propose a new algorithm named TMP-Mine for discovering the TMPs. TMP-Mine uses a special data structure called TMP-Tree to achieve high efficiency in mining process.

5.2. TMP-Tree construction

In order to discover the TMPs efficiently, it is required to construct a TMP-Tree in advance. The purpose of constructing TMP-Tree is to aggregate the temporal movement sequences into the memory in a compact form so that the mining of frequent patterns can be done efficiently. The main advantages of TMP-Tree are (1) only one physical database scan is needed to mine all of the frequent patterns, and (2) the TMP-Tree is compact so that the huge amount of data can be handled efficiently.

Each node in TMP-Tree is termed as *location node* (*LNode*) since it semantically represents the location (i.e. the sensor node) an object traversed. Each LNode of TMP-Tree has the following structure:

$LNode := \{l, c, parent\text{-}link, next\text{-}link, childrentable, ITree\}$

The label of LNode, namely *LLabel*, is stored in the variable l, and the number of traversed times for the LNode is stored in the variable c. The parent-link is a pointer linking to the parent LNode and the next-link is a pointer linking to the next LNode with the same LLabel as that of the current LNode. All of the children LNodes of the current LNode are tabulated in the children table. Each LNode is associated with an *interval tree* named *ITree* for recording the temporal information.

Fig. 2 shows the TMP-Tree construction function. The input to this function is the temporal movement log and the function returns the TMP-Tree after inserting every tuple from the log into the TMP-Tree. In the beginning of the construction, TMP-Tree T is initialized (line 1). Then, the tuples are retrieved from the log one by one (line 2), and the *location path* named *LPath* and *interval path* name *IPath* are extracted from each tuple (line 3 and line 4). Afterwards, the extracted LPath is inserted into the TMP-Tree and the function returns the last LNode of T that corresponds to the last LLabel of the LPath. The function then inserts IPath into the ITree on the returned LNode.

Besides, a LLabel table is maintained together with a TMP-Tree to record the total visited times for each LLabel/sensor node and the *last-inserted-LNode* for each LLabel. The logical structure for each tuple of LLabel table is represented as below:

$LLabel := \{l, c, last\text{-}inserted\text{-}LNode\}$

Fig. 3 shows the function for inserting an LPath into a TMP-Tree with specified count. The function first fetches the root node of T and stores it into a temporary node *lnode*, and LLabel table *lt* (line 1). Then, the function discretizes the LPath into an array and inserts each label into TMP-Tree *in order* (line 2). For a LLabel l, if the *lnode* has a child with label $= l$ (line 3), the children table will be looked up and the entry with label $= l$ is assigned to *lnode* (line 4). The count of *lnode* is also increased by the specified count (line 5). If *lnode* has no child with LLabel $= l$, meaning that the inserting node is a new LNode on the LPath, a new LNode will be created with LLabel $= l$ (line 7) and the count of *lnode* is set as the specified count (line 8). Moreover, the last-inserted-LNode pointer and TMP-Tree structure (line 9–line 11) will be updated so that we can keep track of all LNodes with a specified LLabel via the LLabel table. In the final step, the count of current LLabel in LLabel table is increased (line 13), and the function returns the last LNode on the LPath (line 15).

Fig. 4 shows the procedure for inserting an IPath into the ITree on LNode. The node of ITree is termed as *interval node*, namely *INode*, for it semantically represents the time interval between the traversed sensor nodes. The corresponding label of INode is termed as *interval label* or *ILabel*. Each INode has the following structure:

$INode := \{l, c, parent\text{-}link, peer\text{-}link, childrentable\}$

The insertion procedure is similar to the LPath insertion, except that this procedure uses *peer-link* structure instead of next-link for connecting each INode (line 9). The peer-link is a pointer that points to the next INode

```
Input: the temporal movement log D
Output: TMP-Tree T
Method: TMP_Tree_Construct(D)
1.    T ← Φ
2.    FOREACH tuple τ in D
3.        lp ← ExtractLPath(τ)
4.        ip ← ExtractIPath(τ)
5.        lnode ← InsertLPath(T, lp, 1)
6.        InsertIPath(lnode, ip, 1)
7.    END FOREACH
8.    RETURN T
```

Fig. 2. TMP-Tree construction function.

```
Input: a constructed TMP-Tree T, LPath (Location Path) lp, and the count for the LPath c
Output: the last LNode on lp
Method: InsertLPath(T, lp, c)
1.     lnode ← root(T), lt ← getLLabelTable(T)
2.     FOR i = 1 to length(lp)
3.         IF (getChildren(lnode, lp[i])) ≠ Φ) //the node already exists
4.             lnode ← getChildren(lnode, lp[i]))
5.             Increase the count of lnode by c
6.         ELSE
7.             lnode ← InsertLChild(lnode, lp[i])
8.             set the count of lnode to c
9.             tmpnode ← getLNode(lp[i], lt)
10.            setNextLink(lnode, tmpnode)
11.            Set the last-inserted-LNode to lnode
12.        ENDIF
13.        Increase the count of lp[i] in LLabel table lt by c
14.    END FOR
15.    RETURN lnode
```

Fig. 3. LPath insertion function.

```
Input: LNode (Location Node) lnode, IPath (Interval Path) ip, and the count for the IPath c
Method: InsertIPath(lnode, ip, c)
1.   inode ← ITree_root(lnode), it ← getILabelTable(lnode)
2.   FOR i = 1 to length(ip)
3.       IF (getChildren(inode, ip[i])) ≠ Φ) //the node already exists
4.           inode ← getChildren(inode, ip[i]))
5.           Increase the count of inode by c
6.       ELSE
7.           inode ← InsertIChild(inode, ip[i])
8.           set the count of inode to c
9.           tmpnode ← getINode(i, it)
10.          setPeerLink(inode, tmpnode)
11.          set the last-inserted-LNode to inode
12.      ENDIF
13.      Increase the count of ip[i] in ILabel table it by c
14.  END FOR
```

Fig. 4. IPath insertion function.

on the *same height* of the ITree. By keeping track of the peer-link, it is easy to sum the counts of ILabels on the same height of ITree.

5.3. TMP-Mine algorithm

Fig. 5 shows the detailed algorithm for TMP-Mine, which takes a depth-first search (DFS) approach like WAP-Mine (Pei et al., 2000). While WAP-Mine was designed for mining of single dimensional sequential patterns, the proposed TMP-Mine algorithm can manipulate two-dimensional patterns including location and time attributes simultaneously. The algorithm recursively constructs the TMP-Trees and mines the TMP-Trees until the termination condition is met. First, we list all the LLabels with count greater than the support threshold δ by scanning the LLabel table of current TMP-Tree, and the labels are stored into a temporary set (line 1). If the set is empty (line 2), meaning that no more reconstruction for TMP-Tree is needed, the prefix pattern of current TMP-Tree is output as one of the TMPs (line 3). Otherwise, for each frequent LLabel *llabel*, we fetch the LNodes with LLabel = *llabel* from current TMP-Tree into a temporary set *LNode_tmp* by tracking the last-inserted-LNode pointer (line 5 and line 6).

Under the prefix *llabel*, we accumulate the count of each distinct LLabel from the ancestor LPaths of LNodes in *LNode_tmp* (line 7). The function, getFrequentLIPair( ), returns the set of frequent *location-interval pair* (*LIPair*) where each LIPair in this set is with the count greater than the support threshold δ. We use the set *FreqLIPair* to record the frequent LIPair (line 8). If the *FreqLIPair* is empty (line 9), meaning no more reconstruction is needed, the prefix pattern is output as one of the TMPs and the procedure returns (line 10). Otherwise, we reconstruct the TMP-Trees for each frequent LIPair in FreqLIPair (line 13), and the mining procedure is invoked recursively (line 14) to discover all of the TMPs.

5.4. TMP-Tree reconstruction

As described in Section 5.3, the mining process of TMP-Mine requires recursive reconstruction for the TMP-Trees. Fig. 6 shows the TMP-Tree reconstruction algorithm. The algorithm begins by initializing a TMP-Tree T'(line 1). For each *lnode* in *LNode_tmp* (line 2), we get its *cross-peer nodes* by *iprefix* from the ITree (line 3). In a TMP-Tree, an LLNode and an INode are in cross-peer relation if and only if they are of the same height. Note that the LNodes in *LNode_tmp* have the same LLabel (referred to line 12–line 15 in TMP-Mine algorithm) and the variable *iprefix* is the interval part of the *lipair*. For example, the interval part of a LIPair $(L_a, 10)$ is 10. All of the cross-peer nodes with ILabel = *iprefix* will be returned with the count of them summed up (line 4). Then, the function InsertL-Path( ) is invoked to insert the new LPath with the sum as the count, and the last LNode of the LPath is returned for later insertion (line 5). Afterwards, the IPaths, whose last INode's ILabel is equal to the current ILabel, i.e. *iprefix*, will be inserted into the returned LNode (line 6–line 8). After all of the LNodes in *LNode_tmp* are processed, the function returns the TMP-Tree T' (line 10).

5.5. Temporal movement rules

For a discovered TMP, $P_t = \langle(l_1, i_1, l_2, i_2, \ldots, l_m)\rangle$, the form of the corresponding TMR R_t and the definitions of confidence $conf(P_t)$ and strength $strength(P_t)$ are given as:

$$R_t = \langle(l_1, i_1, l_2, i_2, \ldots, l_{m-1}, i_{m-1})\rangle \rightarrow \langle(l_m)\rangle \quad (1)$$

$$conf(P_t) = \frac{\sup(\langle(l_1, i_1, l_2, i_2, \ldots, l_m)\rangle)}{\sup(\langle(l_1, i_1, l_2, i_2, \ldots, l_{m-1}, i_{m-1})\rangle)} \times 100\% \quad (2)$$

We term the last location of antecedent, namely l_{m-1}, as *LLocation*. Besides, in order to reveal the *strength* of each rule, each rule is ranked by the following formula that considers both of support and confidence:

$$strength(R_t) = \sup(P_t) \times conf(P_t) \quad (3)$$

```
Input: a constructed TMP-Tree T, a specified support $\delta$, and the prefix pattern PfPtn
Output: all of the frequent TMPs
Method: TMP_Mine(T, $\delta$, PfPtn)
 1.  lt ← LabelTable(T)
     FreqL ← getFrequentLabel(lt, $\delta$)
 2.  IF (FreqL ==$\Phi$)
 3.    output prefix pattern PfPtn and RETURN
 4.  ENDIF
 5.  FOREACH llabel in FreqL
 6.    LNode_tmp ← getNodesByLLabel(llabel)
 7.    FreqAncestorLabels ← getFrequentAncestorLabels(LNode_tmp)
 8.    FreqLIPair ← getFrequentLIPair(LNode_tmp, FreqAncestorLabels)
 9.    IF (FreqLIPair ==$\Phi$)
 10.     output prefix pattern PfPtn and RETURN
 11.   ENDIF
 12.   FOREACH lipair in FreqLIPair
 13.     T' ← Reconstruct_TMP_Tree(T', PfPtn, lipair, LNode_tmp)
 14.     newPfPtn ← generate new prefix pattern
         TMP_Mine(T', $\delta$, newPfPtn)
 15.   END FOREACH
 16. END FOREACH
```

Fig. 5. TMP-Mine algorithm.

```
Input: a TMP-Tree T, the prefix pattern PfPtn, the LIPair (Location-Interval Pair) lipair, the
LNodes (Location Node) with the same LLabel (Location Label) LNode_tmp
Output: a reconstructed TMP-Tree T'
Method: Reconstruct_TMP_Tree(T, PfPtn, lipair, LNode_tmp)
 1.  T' ←$\Phi$
 2.  FOREACH lnode in LNode_tmp
 3.    iprefix← getIntervalPart(lipair)
       INode_tmp ← getCrossPeerNodesByILabel(lnode, iprefix)
 4.    c ← sum the count
 5.    lp ← getLPath(lnode)
       endnode ← InsertLPath(T', lp, c)
 6.    FOREACH inode in INode_tmp
 7.      ip ← getIPath(inode), icount ← get the count of the inode
         InsertIPath(endnode, ip, icount)
 8.    END FOREACH
 9.  END FOREACH
 10. RETURN T'
```

Fig. 6. Algorithm for TMP-Tree reconstruction.

Since a large number of rules could be generated, most of traditional data mining methods need a function utilizing hashing tables (Su et al., 2000) or hashing trees (Agrawal and Srikant, 1995) to accelerate the rule access. However, we do not need any accelerating function for accessing the rules. As described in Section 4, the rules will be deployed over the networks based on the location-related criterion. Therefore, dispatching TMRs to sensors by LLocation of each TMR requires only one scan over the physical rule repository. Take the antecedent $\langle(l_1, i_1, l_2, i_2, \ldots, l_{m-1}, i_{m-1})\rangle$ of a TMR for example. Since the LLocation of the antecedent is l_{m-1}, the sensors to load this rule are those within the neighboring radius of l_{m-1}. Considering that the rule that has been dispatched will not be used again in the future, no accelerating function is needed in our application.

In Section 7, we will show through experimental results that ranking rules by strength instead of support or confidence can save more energy. Moreover, if two or more rules have the same strength value, the rule with larger confidence will be given higher priority over other rules.

5.6. An elaborate example

We illustrate the process of discovering TMPs by an elaborate example. Fig. 7 shows the TMP-Tree constructed

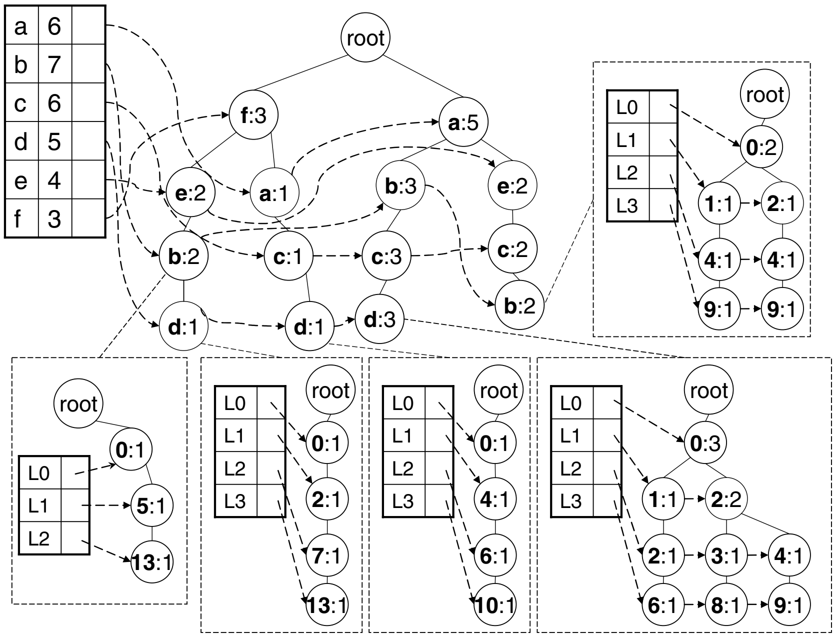


Fig. 7. The TMP-Tree constructed from Table 1.

from the log in Table 1. In this TMP-Tree, each LNode is represented in form of "L:C", where L is the LLabel and C is the count. For illustration, each ITree is surrounded by a dotted rectangle. The representation of an INode is similar to that of an LNode, denoted as "I:C", where I is the ILabel and C is the count associated with this ILabel.

Take the first tuple in Table 1, $\{(a,1)(e,3)(c,5)(b,10)\}$, as example for insertion action. The LPath and IPath extracted from this tuple are $\{(a)(e)(c)(b)\}$ and $\{(1-1)(3-1)(5-1)(10-1)\} = \{(0)(2)(4)(9)\}$, respectively. Because the TMP-Tree is initialized as an empty one, four LNodes for this LPath are created in the current TMP-Tree. These four LLabels are also tabulated in the LLabel table. Besides, a new ITree is constructed with the IPath $\{(0)(2)(4)(9)\}$ on the last LNode that corresponds to this LPath, i.e., the LNode with LLabel $= b$. As stated in Section 5.1, a peer-link structure is maintained to connect the INodes on the same height of the ITree. Hence, the peer-links of the entries in the ILabel table should be set to the corresponding INodes.

In inserting the second tuple, $\{(a,3)(b,5)(c,7)(d,12)\}$, into the TMP-Tree, the first LLabel of the LPath $\{(a)(b)(c)(d)\}$ is a, which is already the child of the root and the count of the LNode is increased by 1 instead of creating a new LNode for this LLabel. On inserting the next LLabel b into the TMP-Tree, a new LNode will be created because the current LNode has only one child labeled as e instead of b. Note that LLabel b has already existed in the LLabel table. Hence, it is required to break the link of last-inserted-LNode of entry b in LLabel table and link it to the newly created LNode. Meanwhile, the next-link of the newly created LNode is pointed to the original LNode, which is the LNode pointed by the original last-inserted-LNode of entry b. By maintaining this linking list, we can keep track of all the LNodes with LLabel $= b$. The same procedure will be performed on the remaining two LLabels, c and d. At the last step, the IPath, $\{(0)(2)(4)(9)\}$, is inserted into the ITree of the last LNode corresponding to LLabel d. The TMP-Tree as shown in Fig. 7 is constructed by inserting the tuples in Table 1 *in order*. Different insertion order will produce different TMP-Tree structures.

Once the TMP-Tree is constructed, we are able to discover the TMPs. Fig. 8 illustrates partial of the mining process. In the beginning, the LLabels with count greater than the support threshold will be fetched after a scanning on the LLabel table. In this example, we set the support threshold as 20%. It is to say that the pattern with support greater than or equal to 2 ($8 \times 20\% = 1.6$) can be one of the TMPs. In Fig. 8a, the count of each LLabel is greater than 2, thus all of the LLabels should be considered in the subsequent process. Then, an arbitrary frequent LLabel is selected as the pattern base for reconstructing the TMP-Tree. If we take the LLabel d as the pattern base for reconstruction, three LPaths, $\{(f)(e)(b)\}$:1, $\{(f)(a)(c)\}$:1 and $\{(a)(b)(c)\}$:3, are obtained as shown in Fig. 8b, where the numbers denote the count of the LPath that ends with LLabel $= d$. Based on the LPaths, the counts of distinct LLabels are accumulated as shown in Fig. 8c. Meanwhile, the LLabel e is pruned because the count of LLabel e is less than the support threshold under the current pattern base $\{(d)\}$.

For each frequent LLabel $llabel_i$, we fetch all the LNodes with LLabel $= llabel_i$ by the last-inserted-LNode link. Each LNode forms several intervals by subtracting the ILabel value of the cross-peer INode from the ILabel value of the base LNode's cross-peer INode. Take the path $\{(f)(e)(b)\}$ for example. The ILabel of b's cross-peer INode is 7 and the ILabel of the root's cross-peer INode, d, is 13.

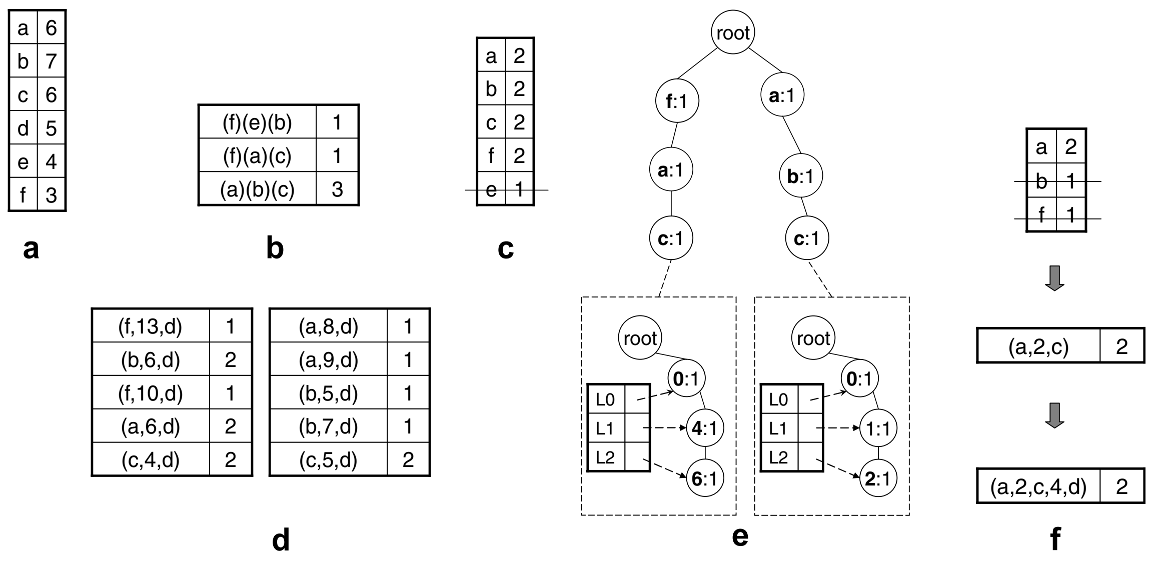


Fig. 8. (a) The LLabel (Location Label) corresponding to the initial TMP-Tree. (b) The LPaths (Location Paths) ending with LLabel = d. (c) The LLabel (Location Label) table under the pattern base set to d. (d) The summary of LIPairs (Location-Interval Pairs) under the pattern base set to d. (e) The TMP-Tree under the pattern base set to $(c,4,d)$. (f) The LLabel table and summary of LIPairs under the pattern base set to $(c,4,d)$, and the discovered TMP.

Table 2
The discovered F-TMPs from Table 1

F-TMPs	F-TMPs
$(a,2,b)$	$(b,1,c)$
$(a,2,c,4,d)$	$(b,6,d)$
$(a,2,\mathrm{c})$	$(c,4,d)$
$(a,4,\mathrm{c})$	$(c,5,b)$
$(a,4,c,5,b)$	$(c,5,d)$
$(a,6,d)$	$(\mathrm{e},8,b)$

The interval between b and d is 6 obtained by subtracting 7 from 13, denoted as (b, 6), which is called a LIPair. The summary of LIPairs is shown in Fig. 8d, in which the LIPairs with count less than 2 have been pruned.

The TMP-Mine algorithm advances by concatenating an arbitrary LIPair with the current prefix pattern as the new pattern base for reconstruction. For example, Fig. 8e shows a reconstructed TMP-Tree if we concatenate $(c,4)$ to (d) as the pattern base. Referring to Fig. 7, is it observed that the both LPaths contain $(c,4,d)$ in the TMP-Tree. Under the pattern base $(c,4,d)$, the algorithm will construct two ITrees for LLabel c. In this TMP-Tree, only LLabel a is frequent. We continue to accumulate the LIPairs and only one LIPair, $(a,2)$, is obtained. Under the pattern base for the reconstructed TMP-Tree, no LLabel has count greater than the support threshold. Therefore, $(a,2,c)$ is concatenated with the pattern base $(c,4,d)$ to form a TMP, as shown in Fig. 8f. By recursively reconstructing and mining the TMP-Trees, all TMPs will be discovered as listed in Table 2.

6. Proposed prediction strategies

In this section, we describe how the discovered TMPs and TMRs are applied to predicting the location of each missing object. For the generated TMRs as described in Section 5, they are deployed over the sensor network by loading the location-related TMRs into corresponding nodes. We propose two prediction strategies, namely PTMP and PES + PTMP, for achieving the prediction tasks. PTMP is a non-velocity-based prediction strategy that exploits the TMRs to predict the location of the missing object, while PES + PTMP is a hybrid strategy that incorporates the well-known velocity-based strategy named PES with PTMP, using both information of detected velocity and the TMRs.

In an OTSN, a location prediction requires two message transmissions in order to know whether the missing object is recovered. The introduced additional communication cost caused by the message transmission is the energy consumed by the transmission and receiving operations between radio components in two sensor nodes and the activation power of two nodes. Hence, it is infeasible to have unlimited predictions in real-time and practical object tracking application. Our pattern-based prediction strategies use the *ranked* TMRs one at a time in predicting the location. For simplicity and generality, the real-time constraint for prediction is represented by *number of predictions* or *TOP-N* predictions in this paper. Hence, a tight real-time constraint corresponds to a low TOP-N value, and a loose constraint corresponds to a higher value contrarily.

Fig. 9 shows the PTMP algorithm for recovering missing objects. The input parameters for this algorithm are the N-gram value, N-gram method, TOP-N value, neighboring radius and ranking method for TMRs. The N-gram method is used to induce the most likely location an object will visit next based on its previous movement behaviour. Two variants, namely standard N-gram and N^+-gram, are considered here. The algorithm begins with composing the antecedent for prediction (line 1). The initial antecedent

```
Input: N-gram value $n$, N-gram method $N_m$, TOP-N constraint $\alpha$, Neighboring radius $r$, and
Ranking method $R$
Output: return whether the object can be found by PTMP
Method: PTMP ($n$, $N_m$, $\alpha$, $r$, $R$)
 1.   $bvr \leftarrow$ Object's historical movement behavior
 2.   FOR i=1 to $r$
 3.      IF $N_m$ = $N^+$-gram method
 4.         call PTMP-$N^+$-gram ($bvr$, $n$, $R$, $\alpha$)
 5.      ELSE IF $N_m$ = N-gram method
 6.         call PTMP-N-gram ($bvr$, $n$, $R$, $\alpha$)
 7.      ENDIF
 8.      IF (the object is recovered)
 9.         RETURN true
10.      ENDIF
11.      subtract the number of predictions from $\alpha$
12.      IF ($\alpha > 0$)
13.         $bvr \leftarrow$ remove the LLocation of $bvr$ and sum the last interval values to form
            the new antecedent for prediction
14.      ELSE
15.         invoke the flooding method to recover the object, and RETURN false
16.      ENDIF
17.   ENDFOR
```

Fig. 9. PTMP algorithm.

is obtained by concatenating the location, arrival time and leaving time of the object. Then, either N^+-*gram* or *N-gram* (as shown in Fig. 11) is invoked to recover the missing object (line 3–line 7). If the object can be recovered by the assigned N-gram method, the OTSN continues to track the object (line 8–line 10). Otherwise, we subtract the number of error predictions from the specified TOP-N value, namely α (line 11). In the end of each round, the algorithm will check the value of α. If α is greater than 0, it means that there is still remaining time for more predictions. Hence, we extend the *neighboring radius* in each round for obtaining more TMRs. The new antecedent is obtained by removing the LLocation of the *bvr* and summing up the last two interval values (line 13). The purpose of regenerating new antecedent is to seek more TMRs for prediction. Take the antecedent $\langle(l_1, i_1, l_2, i_2, \ldots, l_{m-1}, i_{m-1})\rangle$ for example. Suppose that the object is currently at location l_{m-1} and there are no more TMRs for prediction, the antecedent will be modified as $\langle(l_1, i_1, l_2, i_2, \ldots, l_{m-2}, i_{m-2} + i_{m-1})\rangle$, where the LLocation is removed and the last two intervals are summed for seeking more predictions. Note that the flooding method will be invoked to recover the object if the location of object cannot be predicted or no more prediction is allowed (line 15).

Fig. 10 shows the hybrid prediction algorithm named PES + PTMP for recovering objects. The input parameters are the same as those to PTMP and it works as follows. It first uses the latest detected velocity of the object to predict its current location (line 1). If the object can not be recovered by the velocity-based prediction, the algorithm will invoke PTMP to recover the missing object. Here the TOP-N value is subtracted by 1 due to the error prediction that has been made by PES (line 5).

Fig. 11a gives the PTMP-N-gram prediction algorithm. The input parameters for this algorithm are the historical movement behavior of the object, N-gram value and ranking method for TMRs, and it returns if the object is found or not. In the beginning, we extract the last n LIPairs from

```
Input: N-gram value $n$, N-gram method $N_m$, TOP-N constraint $\alpha$, Neighboring radius $r$, and
Ranking method $R$
Output: return whether the object can be predicted by PES+PTMP
Method: PES+PTMP($n$, $N_m$, $\alpha$, $r$, $R$)
 1.   use the latest detected velocity of the object to predict its current location
 2.   IF (the object is found),
 3.      RETURN true
 4.   ELSE
 5.      call PTMP ($n$, $N_m$, $\alpha$-1, $r$, $R$)
 6.   ENDIF
```

Fig. 10. PES + PTMP algorithm.

a

```
Input: Object's historical movement behavior bvr, N-gram value n, and Ranking method R
Output: whether the object can be found by PTMP-N-gram or not
Method: PTMP-N-gram (bvr, n, R, α)
1.  atc ← The last n location-interval pair from bvr
2.  csq ← Get the consequent by the antecedent atc from the TMRs ranked by R
3.  FOR j=1 to min(α, | csq |)
4.      activate sensor s_i and check whether the object is in s_i
5.      if the object is found, RETURN true
6.  ENDFOR
7.  RETURN false
```

b

```
Input: Object's historical movement behavior bvr, N-gram value n, and Ranking method R,
TOP-N constraint α
Output: whether the object can be found by PTMP-N+-gram or not
Method: PTMP-N+-gram (bvr, n, R, α)
1.  FOR i = n down to 1
2.      call PTMP-N-gram (bvr, i, R, α)
        subtract the number of predictions from α
3.      if the object is found, RETURN true
4.      if α <= 0, RETURN false
5.  ENDFOR
6.  RETURN false
```

Fig. 11. (a) PTMP-N-gram algorithm. (b) PTMP-N$^+$-gram algorithm.

the movement behavior to form the antecedent for prediction (line 1). By using the antecedent we obtain a consequent set called *prediction set* from the TMRs, where the predicted locations are ranked by the specified rule ranking method such as support, confidence and strength (line 2). After the prediction set is obtained, the corresponding sensor nodes will be activated one by one to recover the object by the *original node* that lost the object (line 4). Finally, the algorithm returns whether the object is found by PTMP-N-gram or not.

Fig. 11b gives the PTMP-N$^+$-gram prediction algorithm. The spirit of this algorithm is that the predicting by a longer antecedent often produces higher precision than that by a shorter one (Su et al., 2000; Tseng and Lin, 2006). However, the applicability will decrease with the increase in antecedent length (Su et al., 2000; Tseng and Lin, 2006). Therefore, the algorithm starts with *high* N-gram value and decreases the N-gram value after each round for the PTMP-N-gram method (line 2). The activated node must report back to the original node whether the missing object is found or not (line 3). The algorithm terminates only if the object is found or the number of predictions exceeds the specified value (line 4).

7. Experimental evaluation

In this section, we evaluate the performance for the proposed TMP-Mine algorithm by varying the parameters in terms of size of movement log and support threshold. Besides, we evaluate the proposed prediction strategies by measuring the *TEC* and *missing rate* under different time constraints. To select the best ranking method for TMRs, we measured the missing rate by applying support, confidence, or strength to ranking the TMRs. Moreover, the evaluation on variations of PTMP was also discussed. In the object tracking experiments, 80% of the simulated data are used for training to obtain TMRs, and the rest 20% are taken as testing set for object tracking. All of the experiments were conducted on a P4 – 2.4 GHz machine with 1GB main memory. The algorithms and the sensor network simulator are implemented in Java. In the following, we first describe the simulation model and then report the representative results for the conducted experiments.

7.1. Experimental setup

To evaluate the performance of the proposed methods, we implemented a simulator that generates the workload data of an OTSN. Moreover, we conduct an experiment on a real dataset. The details of the simulation model and real dataset will be described in Sections 7.1.1 and 7.1.2, respectively.

7.1.1. Simulation model

Table 3 summarizes the primary parameters used in the simulation model with the default setting. In the base experimental model, the network is modelled as a mesh network with size $|W| = 20 \times 20$, and there are N (defaulted as 10,000) objects in this network. Initially, each object arrives at the network on an arbitrary outer sensor node deploying outside of the sensor network at some time. We assume that the behavior of moving objects in the OTSNs is event-driven instead of randomness completely. Hence, we use two parameters l_e and P_e to model the average

Table 3
Primary parameters for the simulation model

Parameter	Description	Default value
$\|W\|$	$W*W$ nodes of network	20
N	Number of objects in the OTSN	10,000
P_e	Average event probability on each node	0.6
l_e	Average event length	4
F	Average event fan-out	2
P_b	Probability of backward movement	0.1
P_n	Probability of next-node movement	0.18
T	Tracking time for each object (s)	120
I	Average stay time on each node (s)	4
V	Average object velocity (m/s)	15
N_r	Neighboring radius (nodes)	2

length and the event probability, respectively. The length of each event is modelled by Poisson distribution with mean l_e defaulted as 4. The event probability indicates the probability for an object to adhere to a certain event, and it is modelled by Normal distribution with mean P_e (defaulted as 0.6). The events of a node are structured by a tree, in which the fan-out of each node is modelled by Normal distribution with mean F (defaulted as 2). Each object in the network may move by adhering to a certain event or randomly. When an object is in random movement, it will move back by the probability P_b (defaulted as 0.1) or randomly move to other nodes in the hexagon network structure by probability $P_n = (1 - P_b)/(6 - 1)$. The node staying time is modelled by Exponential distribution with mean I (defaulted as 4). The tracking time for each object is set as 120 s. We assume the sensing coverage range is 15 m and the average object velocity is set as 15 m/s. For communications between the sensor nodes and the base stations, we utilize a well-known routing algorithm named shortest path multi-hop as used in Xu et al. (2004). We adopted the Rockwell's WINS node (WINS project) as our basis in simulating the energy consumption. Table 4 lists the energy consumption on WINS nodes (Xu et al., 2004). More detailed power analysis of WINS nodes can be found in Raghunathan et al. (2002), Tseng and Tsui (2004), WINS project. The default value settings for the parameters reflect a reasonable and compact environment for OTSN and mobile systems as in related studies (Eagle and Pentland, 2005; Huang et al., 2003; Lin et al., 2006; Wu et al., 2001; Xu et al., 2004).

7.1.2. Real dataset

To evaluate the practicability of our prediction strategies, we also tested a real dataset from CRAWDAD, which

Table 4
Energy consumption on WINS nodes

Component	Mode	Power (mW)
MCU	Activate	360
MCU	Sleep	0.9
Sensor	Activate	23
Radio	Transmission	720
Radio	Receiving	369

was collected in a project named Reality Mining (Reality Mining Project) by MIT Media Lab for discovering complex social behavior (Eagle and Pentland, 2005). The data collects the behavior of one hundred users during the period between July 19, 2004 and May 5, 2005. In the dataset, the Bluetooth devices are used to represent the location of an object as suggested in Eagle and Pentland (2005) for several practical considerations like weak signals in large buildings. We consider the proximate Bluetooth devices as the location in our experiments. In the raw log, each record is represented by the following attributes: record ID, start time, end time, person ID, and device ID. After a scan on the raw log, a sequence of movement with person ID as the primary key can be obtained. Note that the device ID is the static Bluetooth device ID used to represent the location. The number of distinct Bluetooth devices is 20,794.

7.2. Study on performance of TMP-Mine

In this part of experiments, we investigate the performance of TMP-Mine in terms of execution time by varying the parameters, namely size of movement log and support threshold.

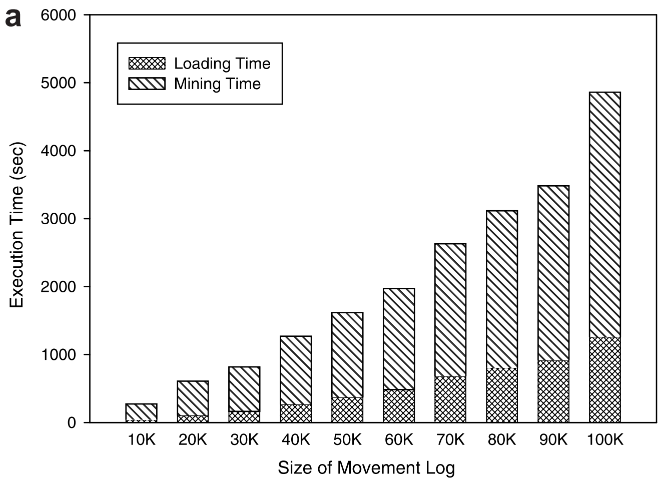
7.2.1. Effects of varying the size of movement log

In this experiment, we evaluate the scalability of TMP-Mine by varying the size of movement log from 10,000 to 100,000 records. We first discuss the execution time, which includes the data loading time and mining time. The *loading time* consists of the time for loading data from hard disk and constructing initial TMP-Tree, while the *mining time* is the time consumed by TMP-Mine algorithm. Obviously, the loading time depends upon the number of log records. As shown in Fig. 12a, the loading time increases approximately linearly with the number of records increased.

Fig. 12b shows that the required memory increases under larger movement log. Under the same parameter setting, the movement behavior in a large dataset is more complex than it is in a small one because there may exist more patterns with support slightly greater than the minimum support in the larger dataset. In other words, the TMP-Tree will have more branches on each node in a larger dataset. Hence, the number of frequent labels and reconstructions are increased. Consequently, the mining time is higher and more memory is required for larger datasets.

7.2.2. Effects of varying the support threshold

This experiment analyzes the impact of varying the support threshold from 0.1% to 1.0%. As stated in Section 5.4, we should reconstruct a TMP-Tree for each label that is greater than the minimum support. As listed in Table 5, the number of TMPs increases with the decrease in support threshold. Therefore, the lower support threshold directly results in more frequent labels and reconstructions, which



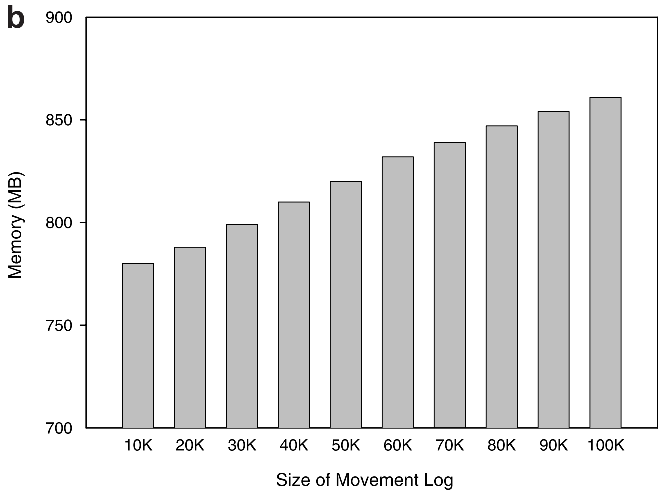
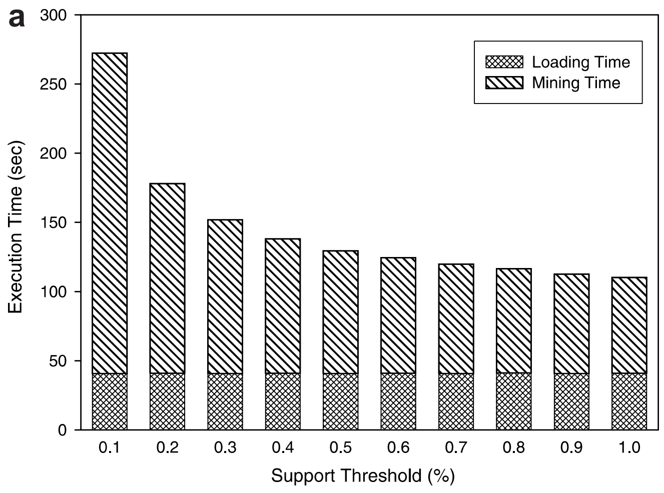


Fig. 12. (a) The execution time vs. log size. (b) The memory requirement vs. log size.



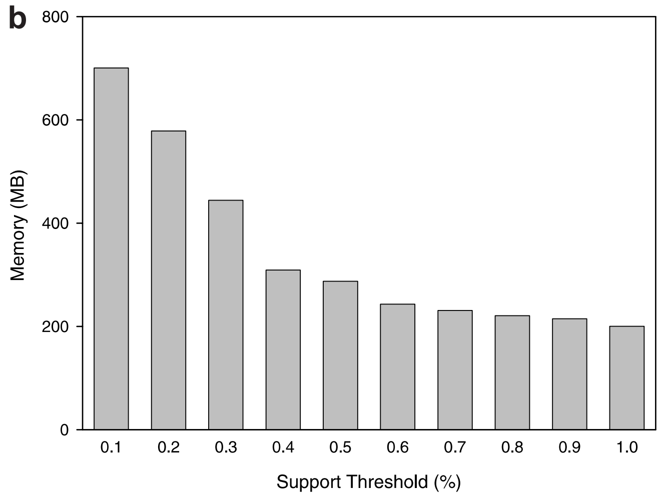


Fig. 13. (a) Execution time vs. support threshold. (b) Memory requirement vs. support threshold.

will increase the mining time as shown in Fig. 13a. As to the loading time, it keeps fixed for different support threshold values since the dataset is invariant. This experiment demonstrates the excellent performance of TMP-Mine even under the low support threshold.

Fig. 13b shows that the required memory increases with the decrease in the support threshold. The reason is that the lower support threshold might have more frequent labels in the TMP-Tree, and the load of TMP-Mine is to allocate more memory to store the frequent labels and the branches.

7.3. Study on performance of prediction strategies

In the following series of experiments, we measure the *TEC* and the missing rate of the proposed prediction strategies. *TEC* indicates the total energy consumed by the OTSN in tracking all objects, and missing rate is the ratio of the error predictions to the total number of movement of objects within a specified deadline. The goal of prediction strategies is to track the moving objects with low *TEC* and low missing rate. Through the performance study on prediction strategies, we use 80% of the simulated data as training set to obtain TMRs, and the rest 20% as testing set for object tracking.

7.3.1. Selection of ranking method

Fig. 14 shows the impact on missing rate when the TMRs are ranked by strength, support and confidence, with the training data occupying 80% of the dataset. It is clear that the strength-ranking approach delivers overall lowest missing rate among the three ranking methods. Moreover, it is observed that the confidence-ranking method has the worst performance in missing rate since

Table 5
Number of TMPs under different support threshold

Sup.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
|TMPs|	19697	5341	2365	1253	743	473	341	233	172	132

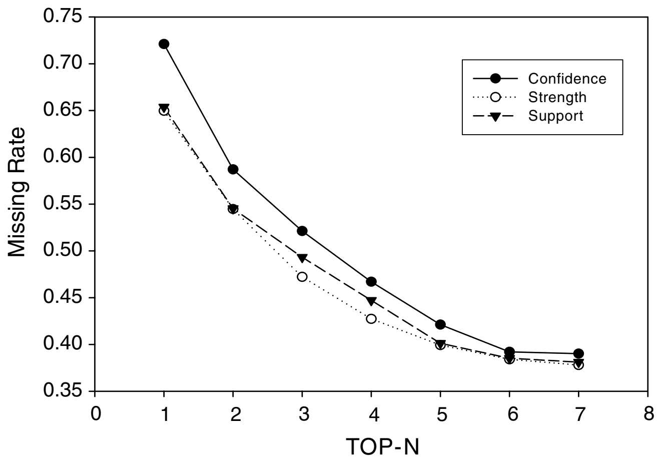


Fig. 14. The missing rate for using strength, support, and confidence to rank the TMRs.

this kind of ranking might recommend a rule with high confidence but very low support. The strength-ranking method considers both the support and confidence of a rule and is demonstrated to have the best performance in terms of missing rate. Therefore, we adopt the strength-ranking method in the subsequent experiments.

In this experiment, we also evaluated the effects of varying the proportion of training set from 50% to 90% of the whole dataset. Although it is observed that the missing rate increases when the proportion of training data is decreased, the mean of standard deviation of the differences under different TOP-N values is only about 1.85%, which is slight as compared with the mean of missing rate (the detailed experimental results are not shown here due to space limitation). This means that the proposed method can still keep robust performance under varied separation ratios for training data vs. testing data. Hence, we set the ratio of training set as 80% of the dataset, which is used popularly in data mining researches (Han and Kamber, 2000).

7.3.2. Performance of variations of PTMP

Fig. 15 shows the performance of PTMP-N-gram and PTMP-N$^+$-gram in terms of *TEC* and missing rate with TOP-N varied from 1 to 7. As shown in Fig. 15a, the *TEC* of PTMP with 1-gram, (denoted as PTMP-1-gram) and PTMP-3$^+$-gram decrease greatly with the increase of TOP-N. Comparatively, the TEC for PTMP-2-gram and PTMP-3-gram decreases much slowly. This phenomenon can be explained by investigating the number of generated TMRs. In our experiments, it is observed that the average number of TMRs stored in each sensor node with length greater or equal to 2 is about 3.56 in average, which is much less than that with length equal to 1 (about 7.70). Therefore, the PTMP-2-gram and PTMP-3-gram will often invoke the flooding recovery for the missing objects due to the few TMRs.

The reason why the *TEC* of PTMP-1-gram decreases greatly with the increase of TOP-N value is that more TMRs are used for prediction. Note that the number of activated sensor nodes by the flooding method is $(6\times1+6\times2+\cdots+6\times m)=6\times(m+m^2)/2$, where the value 6 is the number of neighboring sensors in hexagon network structure and m is the distance (in number of sensors) between the missing object and the original sensor node. Obviously, the energy consumed by the flooding method is much higher than that by our prediction strategies. This explains the phenomenon that the higher TOP-N value is, the lower *TEC* of PTMP-3-gram and PTMP-3$^+$-gram will be. In addition, the observation that the *TEC* of PTMP-3$^+$-gram is always lower than that of PTMP-3-gram is due to the following facts: (1) The prediction set for greater N-gram value has higher priority when we use PTMP-3$^+$-gram for location prediction; (2) Prediction by using a longer antecedent usually produces higher precision than that by a shorter one.

Fig. 15b shows the missing rate for the variations of PTMP method with TOP-N varied from 1 to 7. Although PTMP-3-gram and PTMP-2-gram produce high precision results, the applicability (Su et al., 2000) is low. The low applicability indicates that only limited portions of movement behaviors can be predicted by TMRs. Hence, the missing rate is high and this results in high *TEC*. Besides, it is observed that PTMP-3$^+$-gram have the lowest *TEC* and the lowest missing rate among the four methods. This is because it can dynamically adjust itself to taking advantage of the property of PTMP-N-gram that high precision and high applicability can be achieved under high *N* value and low *N* value, respectively. Hence, we shall use PTMP-3$^+$-gram as the base prediction method in comparisons with other methods in the subsequent experiments. Moreover, it is also observed that the number of TMRs with length greater than or equal to 4 is about 0.67, indicating that setting higher *N* value for PTMP-N$^+$-gram (like PTMP-4$^+$-gram) will not result in significant improvement on *TEC* and missing rate.

7.3.3. Comparisons of different prediction methods

This experiment investigates the performance of different prediction methods in terms of TEC, i.e., the efficiency in energy saving. Four kinds of prediction methods are compared, namely *Continuous Monitoring* (*CM*) (Xu et al., 2004), PES (Xu et al., 2004), PTMP and PES + PTMP. Here, PES + PTMP is a hybrid method by integrating *PES* (*Destination*, *Instant*) method with PTMP. The reason we choose *PES* (*Destination*, *Instant*) for integration is described below. Through our experiments we found *PES* (*Destination*, *Instant*) is the most energy efficient method proposed in Xu et al. (2004). This is because we use parameters P_b and P_n in our model to simulate the activities of objects while not follow the assumption that the object highly intends to move straight forward (Xu et al., 2004), which benefits the other PES variations. The method *PES* (*Destination*, *Instant*) activates only one sensor node per prediction, i.e., the energy penalty is the energy consumed for activating the node when incorrect prediction is made. We also observed that the accuracy

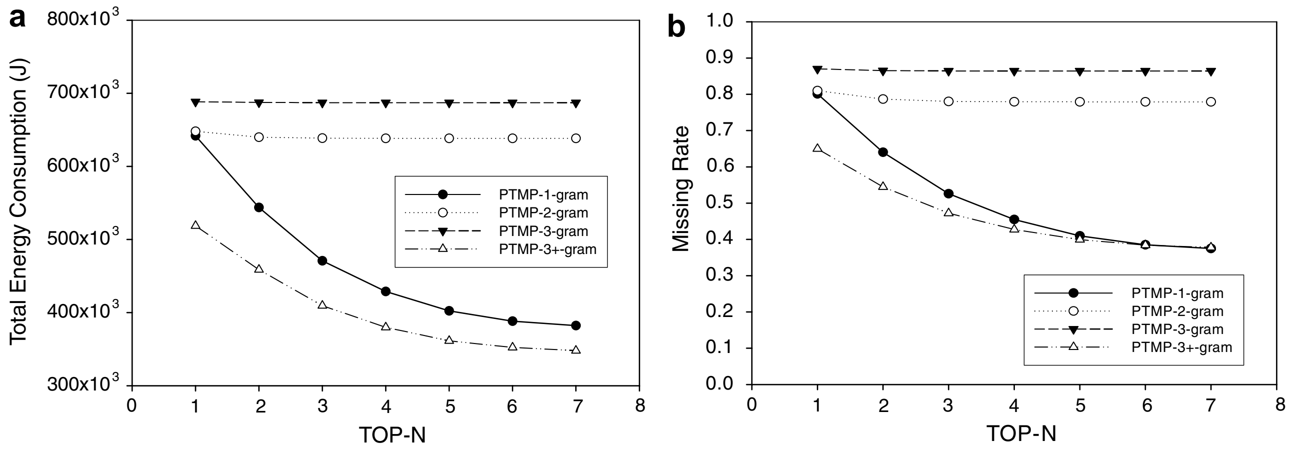


Fig. 15. (a) *TEC* and (b) missing rate for PTMP-N-gram and PTMP-N$^+$-gram with TOP-N value varied.

of the other variations is not absolutely higher than *PES* (*Destination*, *Instant*) but the energy penalty is much higher than it because more than one node will be activated for searching the missing object. We therefore integrate *PES* (*Destination*, *Instant*) with our strategy for comparison. For the PTMP method, we investigate the impacts of varying the support threshold value. Furthermore, we tested different values on parameter X for PES (Destination, Instant) method as well as the hybrid methods.

Fig. 16 shows the experimental results. Note that CM, PES ($X = 0.1$) and PES ($X = 0.5$) are not influenced by varied support threshold, and the *TEC* results for them are shown as PES ($X = 0.1$) > PES ($X = 0.5$) > CM. Recall that in our network model the activated node is scheduled to be in active mode for X seconds and in sleeping mode for ($T - X$) seconds during the T seconds periodically to save the energy. We then explain the phenomenon by the following observations. If an object changes its velocity or moving direction when the corresponding sensor node is in sleeping mode, PES ($X = 0.1$) incurs higher probability in missing the object than PES ($X = 0.5$). Furthermore, since the average velocity in our base model is set as 15 (m/s), the object might move far from the original sensor when the original sensor node is activated again from its sleeping mode. Consequently, the PES with higher X value incurs lower probability in missing objects, indicating that less energy will be consumed because fewer flooding recoveries are performed.

For the *TEC* of PTMP, it is observed that it increases with the increase in the support threshold. In particular, the *TEC* of PTMP is less than that of CM when the support threshold is lower than 0.2. The reason is that PTMP is a pattern-based prediction strategy depending on the TMRs. Fewer TMRs will result in lower applicability. Consequently, PTMP will often apply the flooding method for recovering the missing objects. In contrast, the more TMRs we have, the more energy is required to dispatch the rules over the location-related sensor nodes by data dissemination strategies. In the experiment, we observe that the flooding strategy is invoked considerable times in recovering the missing object due to low applicability. Note that the data dissemination method is only invoked once for dispatching TMRs in the data mining phase. Obviously, the energy penalty of low applicability due to insufficient TMRs is much higher than that of rule dispatching. Hence, the TEC decreases under lower support threshold even though the energy required to dispatch the TMRs is increased. We suggest that the support threshold for PTMP should be set as a small value in order to obtain enough TMRs for enhancing the applicability. In fact, a small support threshold results in only more computation time in the offline mining process and will benefit the online prediction substantially.

For the hybrid methods, namely PES ($X = 0.1$) + PTMP and PES ($X = 0.5$) + PTMP, we observe that either of them has lower *TEC* than those of the pure PTMP or PES. This is because the hybrid strategy exploits both advantages of velocity-based method and event-based method for prediction. That is, the hybrid strategy gets one more chance to predict the location by TMRs instead of immediately using the flooding method to recover the object if PES fails to make correct prediction. Furthermore, we observe that *TEC* of hybrid strategies decrease with the decrease in support threshold. The reason is similar to that for pure PTMP.

7.3.4. Effects of varying the event probability (EP) and TOP-N Value

We explore the impact on missing rate by varying the *EP* and TOP-N values, as shown in Fig. 17. The EP indicates the probability for an object to adhere to some certain event. In Fig. 17, we observe that the missing rate decreases with the increase in TOP-N under a fixed EP. The reason is similar to that as described in Section 7.3.3. We also observe that the missing rate decreases with the decrease in the EP under a fixed TOP-N. Since the higher EP means that an object has higher probability to adhere to a certain event, we get higher probability to obtain correct location by the proposed pattern-based prediction method. Hence, the higher EP results in lower missing rate.

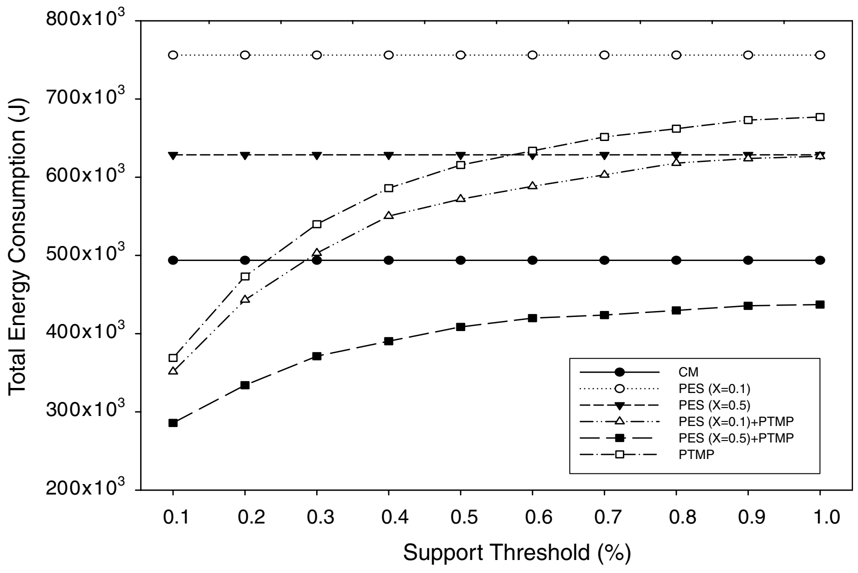


Fig. 16. The *TEC* with support threshold varied for CM, PTMP and PES + PTMP.

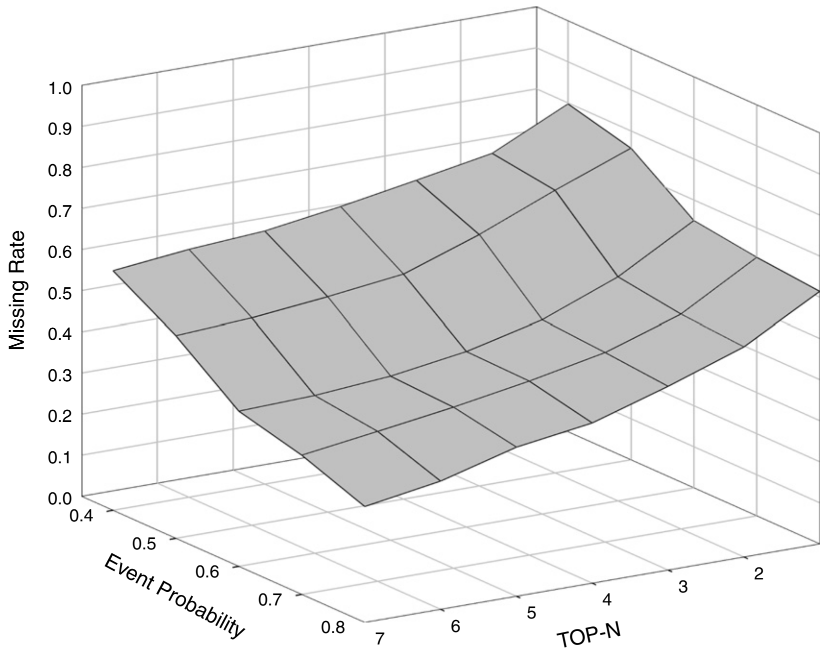


Fig. 17. The missing rate with EP and TOP-N varied for PTMP-3^{+}-gram.

7.3.5. *Effects of varying the object velocity*

In this experiment, we measure the *TEC* with the object velocity varied from 7.5 m/s to 37.5 m/s. Fig. 18a demonstrates that the hybrid strategy PES + PTMP can save more energy than pure PES strategy and PTMP strategy. As the velocity increases, the *TEC* of all of the methods increases. Moreover, we observe the improved ratio in *TEC* for proposed hybrid strategy over PES increases with the increase in the velocity, as listed in Table 6. Here the improved ratio is defined as follows:

$$r = \frac{TEC(PES) - TEC(PES + PTMP)}{TEC(PES)}$$

This can be explained by the following two reasons. First, more energy is required when a sensor node loses an object with higher velocity. This is because the number of nodes activated by the flooding method may also be higher since the object is now far away from the original node. Another reason is that the number of TMRs decreases with the increase in the velocity. We explain the phenomenon by using the density graphs as shown in Fig. 18b. In the density graph of object movement, the gray level of each pixel is normalized by dividing the visited times of nodes by maximum visited times. Note that each object enters the network from the outer nodes. Under the low velocity, the outside nodes of the network will

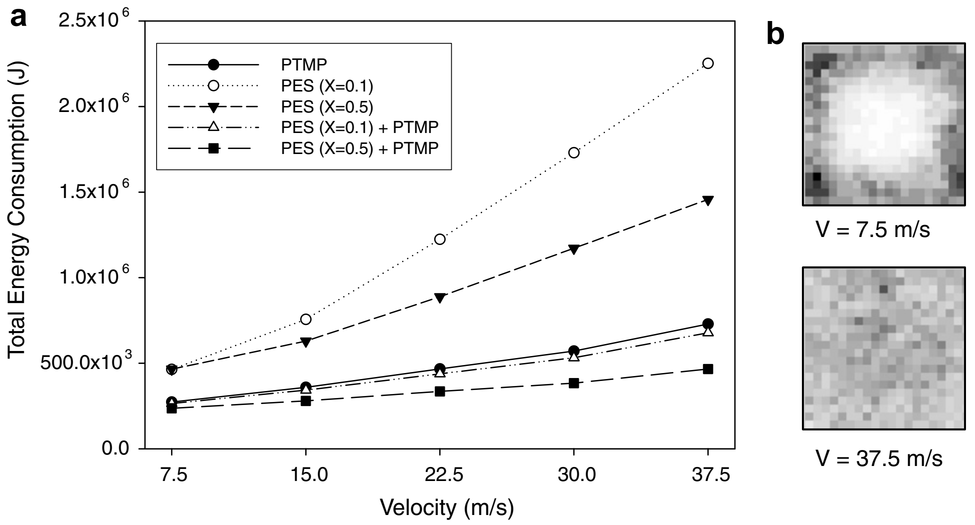


Fig. 18. (a) The *TEC* with average object velocity varied for PES, PTMP and PES + PTMP. (b) The density graph of object movement with the average velocity as 7.5 m/s and 37.5 m/s.

Table 6
The improved ratio on *TEC* for proposed hybrid strategy over PES.

	7.5 m/s	15 m/s	22.5 m/s	30 m/s	37.5 m/s
$X = 0.1$	0.429	0.547	0.642	0.692	0.699
$X = 0.5$	0.492	0.555	0.622	0.673	0.68

attract more visits than the sensors in the inner circle of network. On the contrary, the visits will be dispersed when the average velocity of objects is high. Fig. 18b illustrates the density graphs with the average velocity set as 7.5 m/s and 37.5 m/s, respectively. The higher velocity disperses the visits and results in the decrease of the number of TMRs whose support is greater than the specified threshold. Consequently, fewer number of TMRs results in higher missing rate, which in turn cause more flooding recoveries and higher *TEC*.

7.3.6. Effects of varying the network size

In this experiment, we study the effect of varying the network size from 100×100 (10,000 nodes) to 500×500 (250,000 nodes). Fig. 19 shows that the TEC is increased with the increase in network size, and it becomes stable when the network size is larger than 300×300. As stated in the simulation model, 10,000 objects were generated with the tracking time set as 120 s for each of the five datasets. Hence, fewer TMRs are discovered and the prediction applicability of PTMP becomes lower under a larger network since the objects are more dispersed. Consequently, the TEC is higher under a larger network due to more executions of flooding recovery. This indicates that the performance of PTMP depends on the number of TMRs discovered. However, the performance gain by PTMP will still be significant under a large network if more objects reside in the network such that more TMPs can be discovered.

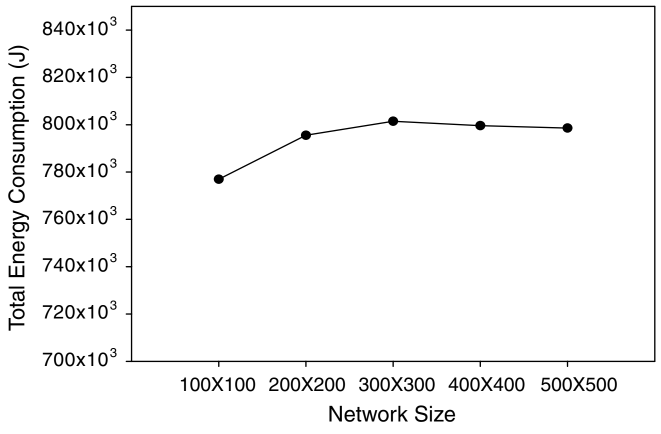


Fig. 19. The TEC with network size varied for PTMP-3^{+}-gram.

7.4. Study on real dataset

For the real dataset as described in Section 7.1.2, we evaluate the performance of our methods in terms of TEC by varying the support threshold and TOP-N values. Before the experiments, we filter out the initial records with date earlier than 1 August, 2004 due to the sparse distribution. Then, the records of the first four months are used as the training set and that of the fifth month (i.e., December, 2004) is used as the testing set. Since the deployment structure of the static Bluetooth devices is unknown, we assume the velocities of objects to be between 1 m/s and 10 m/s that are reasonable for the environment of the dataset. In this way, we can obtain the distance between a person and the node that missed the person by multiplying the missing time with velocity. This information is needed for calculating the recovering energy when the flooding strategy is invoked.

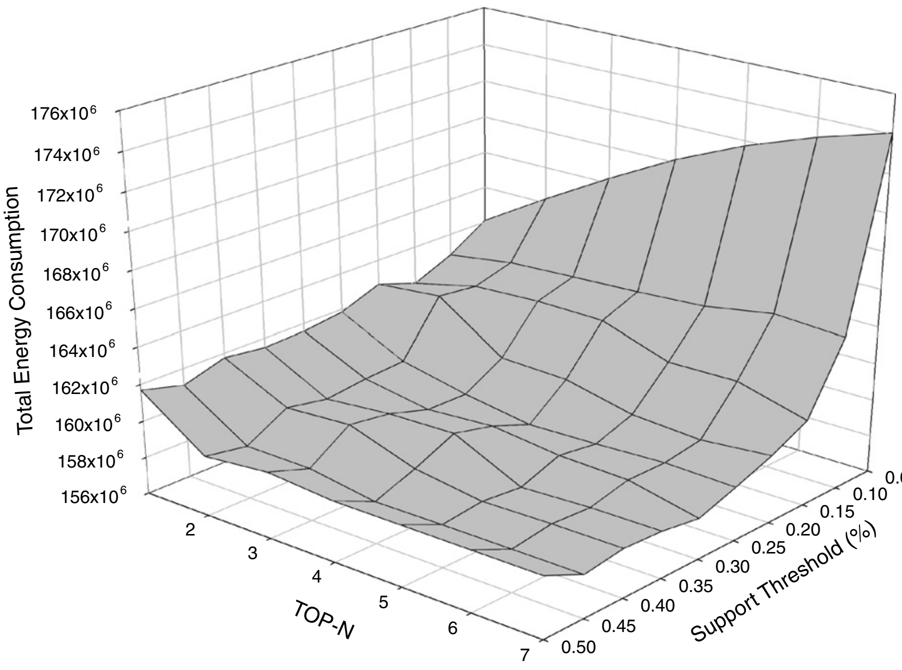


Fig. 20. The TEC of PTMP-3+-gram on the real dataset.

Fig. 20 shows the results for TEC for PTMP-3+-gram with support threshold and TOP-N varied. We observe that the TEC increases dramatically with the increase in TOP-N when the support threshold is lower than 0.10%. On the contrary, the TEC decreases with the increase in TOP-N when the support threshold is higher than 0.10%. The above phenomenon is due to that there exist many noisy TMRs when support threshold is too low. The noisy TMRs will incur the penalty of energy in activating incorrect sensor nodes. Another observation is that the TEC decreases with the increase in support threshold under the same TOP-N constraint. In particular, when the support threshold is lower than 0.25%, the required energy decreases dramatically with the increase in support threshold. The reason is that the number of noisy TMRs is sensitive to the support threshold. Therefore, setting a suitable support threshold is important and it can be determined via experiments. In fact, the experimental results on real dataset are consistent with that of simulated dataset and this also shows the soundness of the simulation model.

7.5. Summary of experimental results

The above experiments consist of two parts: the performance study on TMP-Mine and that of proposed prediction strategies. For the performance of TMP-Mine, it is observed that the TMP-Mine performs well in terms of execution time even under the large dataset (e.g. 100 K) and small threshold (like 0.1%). For the performance of proposed prediction strategies, we first decide the suitable ranking method for TMRs and found that strength-ranking method has the lowest missing rate. For the tracking methods, we found that PTMP-3+-gram performs best in terms of TEC and missing rate under different TOP-N constraints. Moreover, it is observed that the support should be set as a lower value to obtain enough TMRs for benefiting the applicability of the pattern-based strategies. We also observe that integrating PTMP with velocity-based tracking strategy is the more efficient approach if the velocity of objects can be detected. To study the effects of the behavioral characteristics of objects, the parameters EP and object velocity are varied in the experiments and the results show that a higher EP value will reduce the missing rate. Meanwhile, our proposed methods outperform PES, especially when the velocity is high. Finally, the experimental results on real dataset indicate that the energy penalty will occur due to noisy TMRs if the support threshold is too low.

8. Conclusions and future work

In this paper, we have proposed a novel data mining algorithm and the accompanied prediction strategies for tracking the objects in energy efficient manner in OTSNs. The proposed data mining algorithm, namely *TMP-Mine*, can efficiently discover the TMPs for moving objects since only one physical scan on the database is needed. Besides, our study on integrated analysis of both mobility and time interval complements the insufficiency of the past studies that focused on only the aspect of mobility analysis or temporal relationships. To our best knowledge, this is the first work on mining the movement patterns with time intervals in OTSNs. Through empirical evaluation and sensitivity analysis under various system conditions, *TMP-Mine* is shown to perform excellently in terms of execution efficiency and scalability.

In the aspect of prediction strategies, we propose a pattern-based prediction strategy named *PTMP* and a hybrid strategy named *PES* + *PTMP* integrating the PES method with PTMP. The pure pattern-based prediction strategy works with no need to detect the object velocity; hence, it can be applied to the sensor networks with low-end sensor nodes. The hybrid strategy that exploits both the information of object velocity and movement patterns was shown to outperform PTMP and PES in terms of the energy consumption in an OTSN. Therefore, the hybrid strategy serves as an excellent mechanism for OTSNs in which the sensors are equipped with velocity detection ability. To adapt to the limited storage and weak computation ability of sensor nodes, a *rule dispatching mechanism* is also devised by complying the *location-based* criterion. Through experimental evaluation, it is shown that ranking rules by *strength* criteria delivers better results in terms of *TEC* and missing rate than that by using confidence or support.

For the future work, we will apply TMP-Mine on more real datasets and also evaluate the performance of the proposed prediction strategies. Besides, since the discovered TMPs can be exploited in wide applications, we will apply the TMP-Mine method on applications like data dissemination and vehicle monitoring, with the aim to enhance the quality of new applications in sensor networks.

Acknowledgements

This research was partially supported by Ministry of Economic Affairs, R.O.C., under grant no. 92-EC-17-A-02-51-024 and by National Science Council, R.O.C., under grant no. NSC 93-2213-E-006-030.

References

Agrawal, R., Srikant, R., 1995. Mining sequential patterns. In: Proceedings of the 11th International Conference on Data Engineering, pp. 3–14.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. Wireless sensor networks: a survey. Computer Networks 38 (4), 393–422.

Ale, J.M., Rossi, G.H., 2000. An approach to discovering temporal association rules. In: Proceedings of the 2000 ACM Symposium on Applied Computing. pp. 294–300.

Carle, J., Simplot, D., 2004. Energy-efficient area monitoring for sensor networks. IEEE Computer 37 (2), 40–46.

Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., Zhao, J., 2001. Habitat monitoring: application driver for wireless communications technology. In: Proceedings of the First ACM SIGMOMM Workshop on Data Communications in Latin America and the Caribbean.

CRAWDAD Project. http://crawdad.cs.dartmouth.edu/index.php.

Eagle, N., Pentland, A., 2005. Reality mining: sensing complex social systems. Personal and Ubiquitous Computing 10 (#4), 2006.

Goel, S., Imielinski, T., 2001. Prediction-based monitoring in sensor networks: taking lessons from MPEG. ACM Computer Communication Review 31 (5).

Guil, F., Bosch, A., Marin, R., 2004. TSET MAX: an algorithm for mining frequent maximal temporal patterns. In: Proceedings of the Workshop on Temporal Data Mining: Algorithms, Theory and Applications (TDM'04), pp. 71–77.

Han, J., Kamber, M., 2000. Data mining: Concepts and Techniques. Morgan Kaufman Publishers, ISBN 1-55860-489-8.

Hara, T., Murakami, N., Nishio, S., 2004. Replica allocation for correlated data Items in ad hoc sensor networks. SIGMOD Record 33 (1), 38–43.

Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H., 2000. Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences.

Huang, J.L., Chen, M.S., Peng, W.C., 2003. Exploring group mobility for replica data allocation in a mobile environment. In: Proceedings of the ACM International Conference on Information and Knowledge Management, pp. 161–168.

Kyriakakos, M., Hadjiefthymiades, S., Frangiadakis, N., Merakos, L.F., 2003. Multi-user driven path prediction algorithm for mobile computing. In: Proceedings of 14th International Workshop on Database and Expert Systems Applications (DEXA'03), pp. 191–195.

Lee, J.T., Wang, Y.T., 2003. Efficient data mining for calling path patterns in gsm networks. Information Systems 28 (8), 929–948.

Li, Y., Ning, P., Wang, X.S., Jajodia, S., 2003. Discovering calendar-based temporal association rules. Data and Knowledge Engineering 44, 193–218.

Lin, C.Y., Peng, W.C., Tseng, Y.C., 2006. Efficient in-network moving object tracking in wireless sensor networks. IEEE Transaction on Mobile Computing 5 (8).

Mani, M., 2003. Understanding the semantics of sensor data. ACM SIGMOD Record 32 (4).

Padmanabhan, V., Mogul, J., 1996. Using predictive prefetching to improve world wide web latency. ACM Computer Communication Review 26 (3).

Padmanabhan, B., Tuzhilin, A., 1996. Pattern discovery in temporal databases: a temporal logic approach. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 351–354.

Palpanas, T., Mendelzon, A., 1999. Web prefetching using partial match prediction. In: Proceedings of the Fourth Web Caching Workshop.

Pei, J., Han, J., Mortazavi-Asl, B., Zhu, H., 2000. Mining access patterns efficiently from web logs. In: Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining, pp. 396–407.

Raghunathan, V., Schurgers, C., Park, S., Srivastava, M.B., 2002. Energy aware wireless microsensor networks. IEEE Signal Processing Magazine 19 (2), 40–50.

Reality Mining Project. http://reality.media.mit.edu/.

Roddick, J.F., Spiliopoulou, M., 2002. A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and Data Engineering 14 (4), 750–767.

Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A., Chandrakasan, A., 2001. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In: Proceedings of Seventh ACM International Conference on Mobile Computing and Networking (Mobicom'01), pp. 272–287.

Srikant, R., Agrawal, R., 1996. Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the Fifth International Conference on Extending Database Technology (EDBT'06).

Su, Z., Yang, Q., Lu, Y., Zhang, H., 2000. WhatNext: a prediction system for web requests using n-gram sequence models. In: Proceedings of the First International Conference on Web Information Systems and Engineering (WISE'00), pp. 200–207.

Tseng, V.S., Lin, K.W., 2005. Mining temporal moving patterns in object tracking sensor networks. In: Proceedings of the International Workshop on Ubiquitous Data Management (held with ICDE'05), pp. 105–112.

Tseng, V.S., Lin, K.W., 2006. Efficient mining and prediction of user behavior patterns in mobile web systems. Information and Software Technology 48 (6), in press.

Tseng, S.M., Tsui, C.F., 2004. Mining multi-level and location-aware associated service patterns in mobile environments. IEEE Transactions on Systems, Man and Cybernetics: Part B 34 (6).

Tseng, Y.C., Kuo, S.P., Lee, H.W., Huang, C.F., 2004. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. The Computer Journal 47 (4).

WINS project, Rockwell Science Center. http://wins.rsc.rockwell.com.

Woo, A., Culler, D., 2001. A transmission control scheme for media access in sensor networks. In: Proceedings of Seventh ACM Annual International Conference on Mobile Computing and Networking (Mobicom'01), pp. 221–235.

Wu, H.K., Jin, M.H., Horng, J.T., 2001. Personal paging area design based on mobiles moving behaviors. In: Proceedings of IEEE Infocom.

Xu, Y., Winter, J., Lee, W.C., 2004. Prediction-based strategies for energy saving in object tracking sensor networks. In: Proceedings of the Fifth IEEE International Conference on Mobile Data Management (MDM'04), pp. 346–357.

Yang, Q., Li, T., Wang, K., 2004. Building association rule based sequential classifiers for web document prediction. Journal of Data Mining and Knowledge Discovery 8 (3), 253–273.

Yavas, G., Katsaros, D., Ulusoy, Ö., Manolopoulos, Y., 2005. A data mining approach for location prediction in mobile environments. Data and Knowledge Engineering 54 (2).

Ye, W., Heidemann, J., Estrin, D., 2002. An energy-efficient mac protocol for wireless sensor networks. In: Proceedings of the 21st IEEE Infocom, pp. 1567–1576.