An Algorithm for Mining Simultaneous Appearance Patterns of Military Reconnaissance Targets

Yuanguo Cheng and Wenjie Tang
Electronic Engineering College, Navy Engineering University, JieFang Street, Wuhan, China
Email: farcountry@163.com

Abstract. The simultaneous appearance patterns of reconnaissance targets (for short ‘SAPoRT’ latter) which are hided in large history data of the existing military intelligence surveillance and reconnaissance (ISR) system have significant merits on such situations as difficulty target identifying and joint reconnaissance warning. A new algorithm named FTAP-mine for mining SAPoRT was proposed in this paper, which is based on a novel data structure named FTAP-tree and a scheme of conditional search. The FTAP-tree can efficiently store large number of original target information to benefit the following mining process. The method of conditional search can efficiently narrow search space and improve efficiency of the presented algorithm. FTAP-mine employed the idea of pattern growth to recursively fetch frequent prefix patterns from the conditional pattern bases directly, and jointed the suffix to make a pattern growing. Simulation results show that FTAP-mine can efficiently discover frequent SAPoRT in reconnaissance data set and decrease its time complexity and space complexity simultaneously.

1. Introduction
With developing of military ISR system and increasing of various reconnaissance data, it is necessary to employ more efficient data analysis methods in ISR system based on conventional approaches [1, 2]. The SAPoRT which are hided in history reconnaissance data of ISR system have significant merits. For example, they can be employed to identify targets which are difficult to be identified by comparing the detected information about the reconnaissance targets with the trained SAPoRT. Another situation is joint reconnaissance warning. Similar to other methods for mining frequent patterns, the SAPoRT can also be obtained by employing Aprori algorithm, which narrows the candidate frequent item set using a so-called Apriori property [3]. However, if there are $n$ frequent $1$-itemset, the number of candidate frequent $2$-itemset is $\frac{1}{2} n^* (n-1)$.So if the original candidate item set is large, the candidate frequent item set may be more huge. Moreover, this algorithm requires scanning the physical database repeatedly for many times [4, 5]. Aiming at solving these issues, many researchers proposed some improved algorithms based on Apriori. These works mainly focus on improving data storage structure [4, 6] and methods of producing candidate frequent item set [7,8] to enhance time or space efficiency of the proposed algorithms, and so on. Currently, intelligent analysis in some ISR system mainly focuses on signal or information analysis on single specialty [9]. Along with improvement of ISR system, such data mining technologies as association analysis among various reconnaissance targets become more and more necessary. While the history reconnaissance data set is huge, the conventional mining algorithms such as Apriori may lead to tremendous time or space spending.
Based on the above consideration and researches of the predecessor, a novel algorithm to mine the frequent SAPoRT in ISR system was presented. Contributions of this paper include:

1. A novel data structure named FTAP-tree (FTAP-tree: Frequent Target Appearance Pattern tree) has been devised and constructed to aggregate the huge amount of reconnaissance data into memory compactly. A FTAP-tree can reduce memory space efficiently. Moreover, employing a FTAP-tree, the following mining process only needs scan the physical database twice, which heavy reduces scanning times, comparing with the conventional Apriori algorithm.

2. A conditional search method was proposed based on the FP-growth algorithm [4] (FP-growth: to make frequent patterns `grow`). This method can greatly reduce the number of prefix patterns satisfied the search condition, which results in reconstruction of a FTAP-tree is unnecessary and reduces time complexity of the presented algorithm.

3. A novel algorithm for mining SAPoRT in ISR system named FTAP-mine (FTAP-mine: to mine the Frequent Target Appearance Patterns) was proposed, basing on the above FTAP-tree and the search method. The FTAP-mine algorithm adopts the idea of pattern growth, recursively fetches the frequent prefix patterns from the conditional pattern bases directly, and joints the suffix to get a ‘grown’ pattern.

Simulation results show FTAP-mine can efficiently discover SAPoRT and decreases its complexity in time and space simultaneously.

The remainder of this paper is organized as follows: The problem is described in Section 2. We propose our method in Section 3, an efficiency evaluation on FTAP-mine is presented in Section 4. Finally Section 5 addresses a short conclusion and the future work.

2. Problem Formulation

**Definition 1.** Given a reconnaissance target dataset \( D = \{ R_1, R_2, \ldots, R_n \} \), which contains \( m \) targets (denoted by a set \( \{ T_1, T_2, \ldots, T_m \} \)). A SAPoRT (denoted by \( P \)) can be described as a ratiocination relation as \( T_i \land T_j \land \ldots \land T_l \Rightarrow T_j \ (2 \leq j \leq n, 1 \leq i \leq j - 1) \), Where \( T_i, T_j \) represents the \( i \)th target and the \( j \)th target in set \( \{ T_1, T_2, \ldots, T_m \} \), respectively.

**Definition 2.** A support parameter of a pattern \( P (T_i \Rightarrow T_j) \) is defined as
\[
\text{Sup}(P) = \Pr(T_i \Rightarrow T_j) = \left[ \left\{ R_i \in R_j \land R_i \in R_i \land R_i \in R_j \right\} \right] 
\]
which means the simultaneous appearance probability of the target \( T_i \) and the target \( T_j \). Where \( \{ \} \) represents the record number of dataset \( D \). A confidence parameter of a pattern \( P (T_i \Rightarrow T_j) \) is defined as
\[
\text{Con}(P) = \Pr(T_i \Rightarrow T_j | T_i) = \left[ \left\{ R_i \in R_j \land R_i \in R_i \land R_i \in R_j \right\} \right] 
\]
which means condition probability of \( P \) when the target \( T_j \) appears.

**Definition 3.** A pattern \( P \) is said a \( \delta \)-frequent pattern (\( \delta \)-pattern) if \( \text{Sup}(P) > \delta |D| \) where \( \delta \) represents a specified support threshold. If a \( \delta \)-pattern satisfies the condition \( \text{Con}(P) \geq \gamma \) simultaneously, where \( \gamma \) represents a specified confidence threshold, then the pattern \( P \) is call a \( \gamma \)-strong pattern.

According to the above description, the issue can be expressed as how to discover the \( \gamma \)-strong pattern set of \( P \), which meets both \( \text{Sup}(P) > \delta |D| \) and \( \text{Con}(P) \geq \gamma \) in a reconnaissance target dataset \( D \).

3. FTAP-Mine

3.1. Data structure of a FTAP-tree

The original reconnaissance data is large, a compact data structure need to be developed to aggregate the original data into the memory so that the following mining process can be done efficiently. A new structure FTAP-tree is devised to register the original target appearance patterns compactly, which consists of some target nodes and a node index table.

The node of a FTAP-tree is termed as TargetNode for it semantically represents a target that appeared in dataset \( D \). Each TargetNode has the following structure:

\[
\text{struct TargetNode}
\]
{char ID;
    int Count;
    struct TargetNode Next;
    struct TargetNode Child;
}

Where
ID denotes an ID of a target node,
Count is a counter to sum the appearing times of a target node,
Next is a pointer variable that points to the next node with the same ID,
Child is a pointer variable that points to the child nodes of the current node.

For maintaining linkages to traverse prefixes with respect to the same suffix pattern efficiently, a
node index table of all the target nodes in a FTAP-tree is maintained. Each record in this table (termed as
Node-Index) records the appearing times of a target node and the last appearing node with the same
ID. The logic structure of Node-Index is represented as following:
struct Node-Index
    {char ID;
    int Count;
    struct Node-Index Last; }

Where the parameter Last is a pointer that point to the last appearing node with the same ID in a
FTAP-tree, the meanings of other two parameters are same to TargetNode.

3.2. Construction of a FTAP-tree

The algorithm for constructing a FTAP-tree is given in Algorithm 1.

**Algorithm 1**: Constructing a FTAP-tree

**Input**: Reconnaissance data set D

**Output**: A FTAP-tree T

**Method**:
(1) Create a null node root for a FTAP-tree T;
Initialize a pointer variable CurNode;
(2) for (i=1; i<=|D|; i++)
    {if d ∈ D.Record[i] then // D.Record[i]: a set consists of those targets appeared in the ith record.
        {C1 = C1 ∪ {d};
         d.count++};
    for each d∈C1
        {if d.count≥δ|D| then L1 = L1 ∪ {d}};
    Create the target index table NodeIdxTab;
(3) for (i=1; i<=|D|; i++)
    {CurNode= root;
      While (j<length(D.Record[i])-1)
          {CurNode= D.Record[i].Elements[j]
            if CurNode ∈ L1 then FreItemSet[i]= FreItemSet[i] ∪ { CurNode });
            // FreItemSet[i]: the ith frequent item set.
            Arrange the targets of FreItemSet[i] according the orders of NodeIdxTab;
            While (j<length(FreItemSet[i])-1)
                {CurFreTarget= FreItemSet[i].Elements[j]; //CurFreTarget: the current handling target
                    node;
                    if CurFreTarget.ID=CurNode.Child.ID
                        {CurNode=CurFreTarget;
                        CurNode.Count++};
                else
                    {Create a new child node NewChdNode;
                        NewChdNode.Count=1;
                        NewChdNode.ID= CurFreTarget.ID;
CurNode=NewChdNode;}}
(4) for (k=1;k<=NodeIdxTab.MaxRecordNumber;k++)
{CurLinkNode=NodeIdxTab.Record[k];
 Traverse the tree T, express the kth traversed target sets as CurTraNodeSets[k];
For (j=1;j<=CurTraNodeSets[k].Number;j++)
{CurTraNode=CurTraNodeSets[k].Elements[j];
 CurTraNode.Next= CurTraNode;
 CurTraNode.Next.Next= CurTraNodeSets[k].Elements[j-1]}
CurTraNodeSets[k].Elements[j]=null;}

In Algorithm 1, two pointer variable root and CurNode are initialized firstly, the variable root is the root of a FTAP-tree, the variable CurNode is employed to point the current node while constructing a FTAP-tree (step 1). In step 2, Algorithm 1 scans the reconnaissance dataset D and obtains the candidate 1-itemset \((C_1)\) firstly, a frequent 1-itemset \((L_1)\) can be obtained by comparing the parameter ‘Count’ of a target node with the given threshold \(\delta\). Then, a target index table \((NodeIdxTab)\) is created according to a descending sort by the parameter ‘Count’ of each target node. A FTAP-tree is constructed in step 3. Finally, transverse links are built among the nodes with the same ID in a FTAP-tree (step 4).

It can be seen from Algorithm 1 that construction of a FTAP-tree only scans the physical database twice. Once such FTAP-tree is built, all the remaining mining process need to work on the FTAP-tree only, instead of the original reconnaissance database. Moreover, A FTAP-tree can share prefixes, which means more patterns (SAPoRTs) can share a common prefix in a FTAP-tree. Such sharing saves some space for storing patterns and facilitates the support counting of any sub-pattern with the same prefix.

### 3.3. FTAP-MINE

The FTAP-mine algorithm is based on an idea of frequent pattern growth, which was given as Algorithm 2.

**Algorithm 2:** FTAP-mine (mining SAPoRT from a FTAP-tree)

**Input:** a FTAP-tree and a threshold \(\delta\)

**Output:** a SAPoRT set

**Method:**

1. \(FTAPS=\Phi; TmpFTAPS=\Phi\); // FTAPS-SAPoRT set: \(TmpFTAPS\)-the SAPoRT set returned by the function \(ConditionalSearch\);
2. scan the table \(NodeIdxTab\) and obtain an initial pattern set \(iniItemSet\) in which the pattern includes only one node;
3. for each initial pattern \(iniItem\in iniItemSet, do\)
   \[TmpFTAPS= ConditionalSearch (iniItem);\]
   \[FTAPS= FTAPS \cup TmpFTAPS;\]
4. return \(FTAPS\);

Algorithm 2 scans the index table firstly and obtains an initial pattern set in which the pattern includes only one node, these nodes in all initial pattern are taken as frequent 1-itemset, then taking these 1-itemset as initial suffix patterns, the algorithm recursively invokes a conditional search function named \(ConditionalSearch\) to lengthen the suffix patterns until the corresponding prefix pattern base is null, and return a SAPoRT set \((FTAPS)\) finally.

FTAP-mine invokes a function \(ConditionalSearch\) that means instead of searching all SAPoRT once, it turns to search those patterns with the same suffix. The suffix is used as a condition to narrow the search space. As the suffix becomes longer, the remaining search space becomes smaller potentially. The detail of the function \(ConditionalSearch\) was given as Function 1.

**Function 1:** \(ConditionalSearch\): making a pattern grow by searching and joining those nodes with the same suffix on a FTAP-tree

**Input:** an initial suffix pattern \(P_i\)

**Output:** a frequent SAPoRT set obtained from \(P_i\) (named \(FTAPS-P_i\)).
Method:
(1) \{ FTAPS-P_i = \Phi \};
(2) firNode=GetFirstNode(P_i); //firNode: the first node of a sub-pattern P_i;
(3) search all prefix node of the node ‘firNode’ in pattern P_i by tracking the pointer Last in the table NodeIdxTab and the pointer Next of a node, and obtain a prefix node set of P_i, denoted as PrePS-P_i;
(4) prune those nodes whose count is less than \( \delta |D| \), and obtain a frequent prefix node set of P_i, denoted as FrePrePS-P_i;
(5) if FrePrePS-P_i = \Phi return FTAPS-P_i;
(6) else
for each subsidiary pattern Sub-FTAP \( \in \) FrePrePS-P_i, do
\{obtain the last child node which is the parent node of P_i simultaneously, denoted as ParNode-P_i;
while ParNode-P_i.count \( \geq \delta |D| \) )
\{FTAPS-P_i = FTAPS-P_i \cup Sub-FTAP;
FrePrePS-P_i = ConditionalSearch(Sub-FTAP);\} \}

Define the frequent pattern set generated from the suffix pattern P_i as FTAPS-P_i and initialize it null (step 1). Function 1 extracts the first target node from an initial suffix pattern (P_i) (step 2), then searches all prefix nodes of P_i and obtains a set of frequent prefix nodes by pruning those nodes that are not frequent (step 3, step 4). For each subsidiary pattern in the prefix patterns (denoted by FrePrePS-P_i in Function 1), Function 1 obtains its last child node (which is the parent of P_i, simultaneously), then calculates its support value. If it is frequent, adds it to the set of frequent patterns and recursively invokes Function 1 until the set of prefix nodes is null (step 5, step 6).

3.4. An Example
Suppose a simple reconnaissance data set containing 10 records, shown as the following data set:{\{S1,S2,S3,S6,S7\},{S2,S4,S6,S7,S9\},{S1,S2,S3,S5,S6\},{S1,S2,S3,S6\},{S2,S4\},{S2,S4,S6,S7\},{S1,S2,S3,S6,S8\},{S2,S4,S5,S9\},{S4,S6\},{S1,S2,S4,S7\}}. A FTAP-tree corresponded to this data set is shown in figure 1, which is constructed by Algorithm 1.

Suppose the support threshold \( \delta \) is set to 30%, then those nodes whose count is more than or equal to 3 (10*30%=3) are the initial suffix patterns. At beginning, Algorithm 2 scans the index table and takes each node as the initial suffix pattern (i.e. frequent 1-itemset) according to a decreasing support
value, then invokes Function 1 to obtain a candidate 2-itemset, and so on. The final frequent patterns mined by Algorithm 2 are given in table 1.

| Frequent 2-itemset         | Frequent 3-itemset         | Frequent 4-itemset         |
|----------------------------|----------------------------|----------------------------|
| \{S1,S2:5\},\{S1,S3:4\}   | \{S1,S2,S3:4\}             | \{S1,S2,S3,S6:4\}         |
| \{S1,S4:4\},\{S1,S6:4\}   | \{S1,S2,S6:4\}             |                            |
| \{S2,S3:4\},\{S2,S4:5\}   | \{S2,S3,S6:4\}             |                            |
| \{S2,S6:6\},\{S3,S6:4\}   | \{S2,S4,S6:3\}             |                            |
| \{S4,S6:4\},\{S6,S7:3\}   |                            |                            |

Table 1. The frequent SAPORT

Set the confidence threshold \(\gamma\) to be 80%, take the frequent 4-itemset \(\{S1, S2, S3, S6:4\}\) as an example, the corresponding \(\delta\)-pattern and \(\gamma\)-strong pattern are shown in table 2.

| \(\delta\)-pattern       | confidence | \(\gamma\)-strong pattern? |
|--------------------------|------------|----------------------------|
| \(P(S_1 \land S_2 \land S_3 \Rightarrow S_6)\) | 100%       | yes                        |
| \(P(S_1 \land S_2 \land S_6 \Rightarrow S_3)\) | 100%       | yes                        |
| \(P(S_1 \land S_3 \land S_6 \Rightarrow S_2)\) | 100%       | yes                        |
| \(P(S_4 \land S_5 \land S_6 \Rightarrow S_3)\) | 100%       | yes                        |
| \(P(S_1 \land S_2 \Rightarrow S_5 \land S_6)\) | 80%        | yes                        |
| \(P(S_1 \land S_5 \Rightarrow S_2 \land S_6)\) | 100%       | yes                        |
| \(P(S_1 \land S_6 \Rightarrow S_2 \land S_5)\) | 100%       | yes                        |
| \(P(S_2 \land S_3 \Rightarrow S_1 \land S_5)\) | 100%       | yes                        |
| \(P(S_2 \land S_5 \Rightarrow S_1 \land S_3)\) | 66.7%      | no                         |
| \(P(S_3 \Rightarrow S_1 \land S_5 \land S_6)\) | 80%        | yes                        |
| \(P(S_4 \Rightarrow S_1 \land S_2 \land S_6)\) | 50%        | no                         |
| \(P(S_5 \Rightarrow S_1 \land S_2 \land S_6)\) | 80%        | yes                        |
| \(P(S_6 \Rightarrow S_1 \land S_2 \land S_5)\) | 66.7%      | no                         |

Table 2. The mined patterns from frequent 4-itemset

4. Evaluation
Employ Matlab R2014a to generate about 200 thousands of records in which there are 9 reconnaissance targets. The number of targets contained in each record follows a normal distribution whose mean \((\mu)\) is 4 and variance \((\sigma^2)\) is 1. The classical Apriori algorithm was employed to compare the efficiency of FTAP-mine. We conducted 12 troops of experiments to evaluate the running (convergence) time by varying number of reconnaissance records and support threshold \(\delta\), respectively. The experiment results are shown in figure 2. Another experiment was accomplished to evaluate the discovered pattern amount in only FTAP-mine algorithm, whose results were shown in figure 3. All of the experiments were conducted on a PC with dual core Pentium G2030 CPU and 3GB main memory.
Simulation results show that in both algorithms the running time increases almost linearly by increasing number of original records or decreasing support threshold from 30% to 10%, which demonstrates that both methods have perfect extendibility. However all experiments show that FTAP-mine has better capability in time complexity. Similarly, in figure 3 the amount of discovered patterns
by FTAP-mine changes linearly with changing number of original records or support threshold, which further demonstrates the extendibility and the feasibility of FTAP-mine.

5. Conclusion
A new mining method named FTAP-mine to discover SAPoRT in military ISR system was proposed in this paper, which is based on the presented data structure FTAP-tree and a conditional research function. More in-depth researches contain more pattern definitions, more effective storage structure and more mining algorithms to efficiently mine the frequent patterns from the reconnaissance data.

6. References
[1] Yang J B, Xiao D and Guo X. Research on data mining based on net-centric operations. Networks and Information Engineering, 2015.03. pp 96-99. (in Chinese)
[2] Guo J G and Huang S. Study on architecture of big data based on military intelligence analysis and service system. Journal of CAEIT, 2017, 12(4). pp 389-393,413. (in Chinese)
[3] Agrawal R, Imielinski T and Wami A. Mining association rules between sets of items in large databases. ACM sigmod record, 1993, 22(2).pp201-216.
[4] Fan M and Meng X F. Data Mining: Concepts and Technologies, Beijing: China Mechine Press, 2012.01. pp 56 - 159.
[5] Ying G, Zhang Q and Chen L J. Research on application of FP tree based association rule mining on test sequence in train control system, Proceedings of the 36th Chinese Control Conference, DaLian: Dalian University of Technology Press, 2017.07. Pp9993-9997.
[6] Liu Y and Li S. An improved Apriori minim algorithm based on weight [J]. Journal of Xi’an University of Posts and Telecommunications, 2017,22(4).pp95-100.(in Chinese)
[7] Huang K and Wu Y J. An algorithm of mining high utility itemsets with vertical structures. Journal of Dalian University of Technology, 2017.09.57(5). pp 524-530. (in Chinese)
[8] Liu Y L, Xie P and He Q S. A new method based on association rules mining and geo-filter for mining spatial association Knowledge. Chinese Geographical Science, 2017, 27(3). pp 389 - 401.
[9] Pei Z L, Jin P J and Zhang S J. Association rules minim model of radar countermeasure reconnaissance data [J]. Aerospace Electronic Warfare, 2007, 24(4). pp 35-37. (in Chinese)