On The Mining and Usage of Movement Patterns in Large Traffic Networks

Mohammed Al-Zeyadi
Department of Computer Science
University of Liverpool
Liverpool, UK
m.g.a.al-zeyadi@liv.ac.uk

Frans Coenen
Department of Computer Science
University of Liverpool
Liverpool, UK
Coenen@liv.ac.uk

Alexei Lisitsa
Department of Computer Science
University of Liverpool
Liverpool, UK
lisitsa@liv.ac.uk

***Abstract*—Paper presents the Shape Movement Pattern (ShaMP) algorithm, an algorithm for extracting Movement Patterns (MPs) from network data, and a prediction mechanism whereby the identified MPs can be used to predict the nature of movement in a previously unseen network. The principal advantage offered by ShaMP is that it lends itself to parallelisation. The reported evaluation was conducted using both Massage Pass Interface (MPI) and Hadoop/MapReduce; and artificially generated and real life networks. The later extracted from the UK Cattle tracking Systems (CTS) in operation in Great Britain (GB). The evaluation indicates that very successful results can be produced, average precision, recall and F1 values of 0.965, 0.919 and 0.941 were recoded respectively.**

***Keywords*-Big Network Data; Pattern Mining; Movement Patterns; Prediction; Hadoop; MPJ Express.**

I. Introduction

Over the last decade "big network" mining has attracted considerable attention because of the large amount of network data that is increasingly generated by a great variety of application domains: social networks [1], computer networks [2], Peer-to-Peer Networks [3], road traffic networks [4] and so on. The mining of such networks can take different forms, for example we might wish to identify communities within such networks [5], or alternatively we might wish to identify "influencers" within such networks [6]. In this paper we are interested in providing answers to questions of the form: "given the nature and volume of 'traffic' in a given network N at time t what will the nature and volume of traffic be at time $t + \delta$?", or alternatively "given the nature and volume of 'traffic' in a given large network N what will be the nature and volume of traffic in a closely related network M?". Note that this is distinct from other applications that involve movement such as trajectory or link prediction.

The fundamental idea presented in this paper is the usage of the concept of Movement Patterns (MPs) to provide an answer to the above form. A MP is a tuple of the from $\langle F, E, T \rangle$ where F and T are descriptions of two vertices in a network, and E is a description of the movement (traffic) between these two vertices (assuming such movement exists). The objective is thus to extract MPs from a given network where the traffic is known and apply these patterns to a related network where the traffic is unknown. In other words, we wish to apply our MPs to a network work where we only have descriptions of the vertices and not the edges; it is the edges and their nature that we wish to predict.

The challenge, given the above, is the size of the networks (in the evaluation considered later in this paper networks with 25 million edges are used). The first contribution of this paper is the Shape Movement Pattern (ShaMP) mining algorithm. The algorithm leverages knowledge of the restrictions imposed by the nature of traffic patterns to obtain efficiency advantages not available to more traditional pattern mining algorithms that might otherwise be adopted. A further advantage of ShaMP is that it readily lends itself to parallelisation. The algorithm is fully described and its operation analysed using a Linux cluster and two well known distributed computing techniques: (i) Map Reduce (MR) on a top of Hadoop [7] and (ii) Massage Passing Interface (MPI) [8]. The second contribution is a mechanism for applying the identified MPs in related networks so as to predict the nature of the traffic in these networks.

II. Literature Review

In the era of big data the prevalence of networks of all kinds has growing dramatically. Coinciding with this growth is a corresponding desire to analyse (mine) such networks, typically with a view to some social and/or economic gain. A common application is the extraction of customer buying patterns [9], [10] to support business decision making. Network mining is more focused, but akin to graph mining [11]. Network mining can take many forms; but the idea presented in this paper is the extraction of Movement Patterns (MPs) from movement networks. The concept of Movement Pattern Mining (MPM) as conceived of here, to the best knowledge of the authors, has not been previously addressed in the literature. However, patten mining in general has been extensively studied. This section thus presents some background to the work presented in this paper. The section commences with a brief review of the pattern mining context with respect to large networks, then continues with consideration of current work on frequent movement patterns and the impact of distributed processing to find such patterns in large data networks.

A central research theme within the domain of data mining has been the discovery of patterns in data. The earliest examples are the Frequent Pattern Mining (FPM) algorithms proposed in the early 1990s [12]. The main objective being to

discover sets of attribute-value pairings that occur frequently. The most well established approach to FPM is the Apriori approach presented in [12]. A frequently quoted disadvantage of FPM is the significant computation time required to generate large numbers of patterns (many of which may not be relevant). The MPM concept presented in this paper shares some similarities with the concept of frequent pattern mining. However, the distinction between movement patterns and traditional frequent patterns is that movement patterns are more prescriptive, as will become apparent from the following section; they have three parts: (i) sender, (ii) details of movement and (iii) receiver; none of these parts can be empty. Note also that the movement patterns of interest with respect to this paper are traffic movement patterns and not the patterns associated with the video surveillance of individuals, animals or road traffic; a domain where the term "movement pattern" is also used. To the best knowledge of the authors there has been very little (no?) work on movement patterns as conceptualised in this paper.

The MPM concept also has some overlap with link prediction as found in social network analysis where we wish to predict whether two vertices, representing individuals in a social network, are likely to be linked at some time in the future. MPM also has some similarity with the problem of inferring missing links in incomplete networks. The distinction is that link prediction and missing link resolution is typically conducted according to graph structure, dynamic in the case of link prediction [13] and static in the case of missing link resolution [14].

Many effective single-machine FPM algorithms have been proposed typically founded on the concept of some form of set enumeration tree structure [15], [16], that do not readily lend themselves to parallelisation; although some parallel variations of such algorithms have been proposed whereby the tree structures used can be partitioned (see for example [17]). Researchers have thus been motivated to consider *MapReduce* style distributed memory systems [18], [19], and *Message Passing Interface* (MPI) style algorithms [20], [21] for large scale FPM. In the context of the proposed ShaMP algorithm presented later in this paper, the usage of both MPI and MapReduce are considered. More specifically MPJ Express, an implementation of MPI using the Java programming language; and MapReduce on a top of Hadoop, the most popular MapReduce open-source implementation.

III. Problem Statement and Formalism

In the context of the work presented in this paper a network G is defined in terms of a tuple of the form $\langle V, E \rangle$, where V is a set of vertices and E is a set of edges [22]. The vertices can represent individuals (as in the case of social networks), inanimate entities (as in the case of computer networks) or locations (as in the case of distribution and road traffic networks). The edges then indicate connections between vertices (virtual or actual). These edges might be indicative of some relationship, such as a friend relationship, as in the case of social networks; or a "hard" connection as in the case of a wired computer network or a road traffic network. In this paper we conceive of edges as bearing attributes indicative; for example: (i) the number of messages sent from one individual to another in a social network, (ii) the volume of data exchanges between two computers in a computer network, (iii) the quantity of goods sent in a distribution network or (iv) the amount of traffic flow from one location to another in a road traffic network. As such edges are directed (not necessarily the case in other forms of network). To distinguish the networks of interest with respect to this paper from other forms of network considered in the literature we will use the term *movement network*; a network $G(V, E)$.

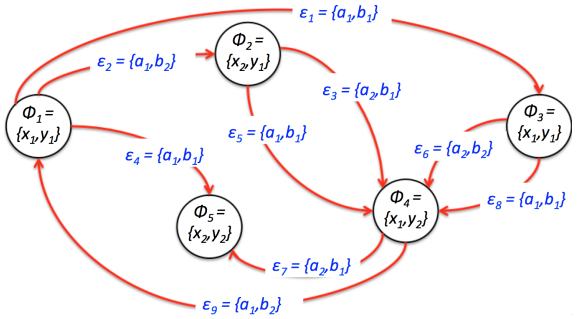

Fig. 1: Example movement network ($V = \{\phi_1, \phi_2, \ldots, \phi_5\}$ and $E = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_9\}$).

An example movement network is given in Figure 1 where $V = \{\phi_1, \phi_2, \ldots, \phi_5\}$ and $E = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_9\}$. Note that some of the vertices featured double as both receiver "to" and sender "from" vertices. Thus the set of from vertices and the set of to vertices are not disjoint. Note also that some vertices in the figure are connected by more than one edge. This is because we do not simply wish to consider traffic flow in a binary context (whether it exists or does not exist) but in terms of the nature of the traffic flow. Thus where vertices in the figure are connected by more than one edge this indicates more than one kind of traffic flow. For example in a distribution movement network two edges connecting one vertex to another might indicate the dispatch of two different commodities. As such edges have a set of attributes associated with them A_E. Similarly the vertices have a set of attributes associated with them, A_V. The nature of these attribute sets will depend on the application domain, however each attribute will have two or more values associated with them. Where necessary we indicate a particular value j belonging to attribute i using the notation v_{i_j}.

As established in the introduction to this paper, given a large traffic network, of the form shown in Figure 1, the nature of the traffic exchange between vertices can be described in terms of *Movement Patterns* (MPs); three part patterns describing some traffic exchange comprising: (i) a from vertex (F), (ii) the nature of the traffic (E) and (iii) a to vertex (T). Thus "sender", "details of movement" and "receiver". A movement pattern can thus be conceived of as a tuple of the form $\langle F, E, T \rangle$, where F and T subscribe to the attribute set A_V, and E to the attribute set A_E. In the figure $A_V = \{X, Y\}$ and

$A_E = \{A, B\}$; each attribute X, Y, A and B has some value set associated with it, in the figure the value sets are $\{x_1, x_2\}$, $\{y_1, y_2\}$, $\{a_1, a_2\}$ and $\{b_1, b_2\}$ respectively. Given a movement network G we wish to find the set of MPs M in G so that they can be used to predict the set of movement patterns $M\prime$ in a previously unseen (initially unconnected) graph G' comprised of a set of vertices V' that subscribe to the same vertex attribute set A_V. The aim is to predicted the nature of the traffic in the new network.

From the foregoing an MP comprises a tuple of the form $\langle F, E, T\rangle$ where F, E and T are sets of attribute values. The minimum number of attribute values in each part must be at least one. The maximum number of values depends on the size of the attribute sets to which F, E and T subscribe, although an MP can only feature a maximum of one value per subscribed attribute.

The movement networks from which we wish to extract MPs can also be conceived of as comprising $\langle F, E, T\rangle$ tuples. The movement network presented in Figure 1 can thus be presented in tabular form as shown in Figure 2 (for ease of understanding the rows are ordered according to the edge identifiers used in the figure). We refer to such data as "FET" data . Thus MPM can be simplistically thought of as the process of extracting a set M of frequently occurring MPs from a dataset D so as to build a model of D that can then be used to predict traffic in some previously unseen network D' (G') comprised solely of vertices (no known edges, the edges are what we wish to predict). An MP is said to be frequent, as in the case of traditional FPM [23], if its occurrence count in D is in excess of some threshold σ expressed as a proportion of the total number of FETs in D. With reference to the movement network given in Figure 1, and assuming $\sigma = 30\%$, the set of MPS, M, will be as listed in Figure 3 (the numbers indicated by the $\#$ are occurrence counts).

$$
\begin{array}{l}
\langle \{x_1, y_1\}, \{a_1, b_1\}, \{x_1, y_1\}\rangle \\
\langle \{x_1, y_1\}, \{a_1, b_2\}, \{x_2, y_1\}\rangle \\
\langle \{x_2, y_1\}, \{a_2, b_1\}, \{x_1, y_2\}\rangle \\
\langle \{x_1, y_1\}, \{a_1, b_1\}, \{x_2, y_2\}\rangle \\
\langle \{x_2, y_1\}, \{a_1, b_1\}, \{x_1, y_2\}\rangle \\
\langle \{x_1, y_1\}, \{a_2, b_2\}, \{x_1, y_2\}\rangle \\
\langle \{x_1, y_1\}, \{a_1, b_1\}, \{x_1, y_1\}\rangle \\
\langle \{x_1, y_2\}, \{a_2, b_1\}, \{x_2, y_2\}\rangle \\
\langle \{x_1, y_2\}, \{a_1, b_2\}, \{x_1, y_1\}\rangle
\end{array}
$$

Fig. 2: Movement network from Figure 1 presented in tabular form.

IV. Movement Pattern Mining Using The ShaMP Algorithm

From the foregoing the ShaMP algorithm, as the name suggests, is founded on the concept of "shapes". A shape is a MP template with a particular configuration of attributes

$$
\begin{array}{ll}
\langle \{x_1\}, \{a_1\}, \{x_1\}\rangle & \#3 \\
\langle \{y_1\}, \{a_1, b_1\}, \{x_1\}\rangle & \#3 \\
\langle \{y_1\}, \{a_1, b_1\}, \{y_2\}\rangle & \#3 \\
\langle \{y_1\}, \{a_1\}, \{x_1\}\rangle & \#3 \\
\langle \{y_1\}, \{a_1\}, \{y_2\}\rangle & \#3 \\
\langle \{y_1\}, \{b_1\}, \{x_1\}\rangle & \#4 \\
\langle \{y_1\}, \{b_1\}, \{x_1, y_2\}\rangle & \#3 \\
\langle \{y_1\}, \{b_1\}, \{y_2\}\rangle & \#4
\end{array}
$$

Fig. 3: The MPs (the set M) extracted from the Movement network given in Figure 1 using $\sigma = 30\%$.

taken from A_L and A_E (note, shapes do not specify particular attribute value combinations). The total number of shapes that can exist in a FET dataset D can be calculated using Equation 1, where: (i) $|A_L|$ is the size of the attribute set A_L and (ii) $|A_E|$ is the size of the attribute set A_E. Recall that attributes for F and T are drawn from the same domain. Thus if $|A_L| = 2$ and $|A_E| = 2$, as in the case of the network given in Figure 1, there will be $(2^2 - 1) \times (2^2 - 1) \times (2^2 - 1) = 3 \times 3 \times 3 = 27$ shapes. If we increase $|A_E|$ to 5 there will be $(2^2 - 1) \times (2^5 - 1) \times (2^2 - 1) = 3 \times 31 \times 3 = 279$ different shapes. Thus the number of shapes to be considered increases exponentially with $|A_V|$ and $|A_E|$, and consequently the efficiency of the ShaMP algorithm is directly related to the number of different shapes to be considered.

$$(2^{|A_V|} - 1) \times (2^{|A_E|} - 1) \times (2^{|A_V|} - 1) \qquad (1)$$

In this paper two implementations of the ShaMP Algorithm are considered, one using the MPJ open source Java message passing library and one using Map-Reduce (MR) over Hadoop; the two variations are referred to as ShaMP MPJ and ShaMP Hadoop. The pseudo code for the two algorithms is presented in Algorithms 1 and 2. At a high level both algorithms operate in a similar manner. The input in both cases is a collection of "FETs" $D = \{r_1, r_2, \dots\}$ describing some movement network G and a support threshold σ. The output in both cases is a set M of frequent MPs together with their support counts, $M = \{\langle MP_1, count_1\rangle, \langle MP_1, count_1\rangle, \dots\}$. Both algorithms commence by generating a set of shapes and then proceed by populating the shapes by looping over the input data.

Returning to Algorithm 1 (ShaMP MPJ) the Algorithm first generates the complete set of shapes $ShapeSet$. The algorithm then divides the $ShapeSet$ equally into N subsets according to the number of machines available. Note that all machines work in parallel. For each shape the algorithm loops through the data set D and compares each record $r_j \in D$ with the current shape $shape_i$. A record r_j matches a $shape_i$ if all the attributes featured in $shape_i$ are also featured in r_j (r_j may include additional attributes not featured in $shape_i$). Where a match is found the relevant attribute values in r_i form a MP. If the identified MP is already contained in M we simply update

the support value, otherwise we add the newly discovered MP to M with a support value of 1. Once all shapes have been processed we loop through M and remove all MPs whose support count is less than σ.

Input:
D = Collection of FETs $\{r1, r2, ...\}$ describing network G
σ = Support threshold
Output:
M = Set of frequently occurring MPs
$\{\langle MP_1, count_1\rangle, \langle MP_2, count_2\rangle, \dots\}$
$ShapeSet$ = the set of possible shapes
$\{shape_1, shape_2, \dots\}$
M = $\emptyset$
N = Number of Machines in a cluster
Split And Distributed $ShapeSet$ across a cluster of machines where;
$ShapeSet$ =
$SubShapeSet_1 \sqcup SubShapeSet_2 ... \sqcup SubShapeSet_N$
In Parallel: process all $SubShapeSet$ independently
forall the $shape_i \in SubShapeSet$ **do**
  forall the $r_i \in D$ **do**
    if r_i *matches* $shape_i$ **then** MP_k = MP extracted from r_i
    if MP_k *in* M **then** increment support
    else $M = M \bigcup \langle MP_k, 1\rangle$
  end
end
forall the $MP_i \in M$ **do**
  if *count for* $MP_i < \sigma$ **then** remove MP_i from M
end

Algorithm 1: ShaMP MPJ Algorithm

Input:
D = Collection of FETs $\{r1, r2, ...\}$ describing network G
σ = Support threshold
Output:
M = Set of frequently occurring MPs
$\{\langle MP_1, count_1\rangle, \langle MP_2, count_2\rangle, \dots\}$
Map Function:
Begin:
S = $\emptyset$
$ShapeSet$ = The set of possible shapes
$\{shape_1, shape_2, \dots\}$
forall the $shape_i \in ShapeSet$ **do**
  forall the $r_i \in D$ **do**
    MP_k = MP extracted from r_i
    MP_kSupportCount = Count (MP_k,1)
  end
end
$S = S \bigcup \langle MP_k, 1\rangle$

Reduce Function:
Begin:
Input:
S set from Map Function
Output:
M = Set of frequently occurring MPs M = $\{\langle MP_1, count_1\rangle, \langle MP_2, count_2\rangle, \dots\}$
forall the $\langle MPs, 1\rangle$ *pairs* $\in S$ **do**
  $\langle MPs, count\rangle = Aggregation$ all $\langle MPs, counter\rangle$ pairs that have same MP
  if *count for* $MP > \sigma$ **then** $M = M \cup \langle MPs, count\rangle$
end

Algorithm 2: ShaMP Haddop Algorithm

Referring to Algorithm 2 (ShaMP Hadoop), as in all Map-reduce algorithms, this comprises both Map and Reduce functions. The input for the Map function is again a network G represented in terms of a FET dataset D, and a desired support threshold σ. While the output is a set S, which is a temporary set to hold all $\langle Key, Value\rangle$ pairs; where Key is an MP and the Value is occurrence of MP which at this stage will equate to 1. The input to the Reduce function is the set S and the output is the desired set of MPs, M. The Map task commences by generating the available set of shapes. For each shape, the algorithm loops through D comparing each record $r_j \in D$ with the current shape $shape_i$, where a match is found, a key-value $\langle MP, 1\rangle$ is created. The Reduce function aggregates all the MPs with the same key (MP) so as to produce frequency counts for each key (MP). After that the set S is processed and any MPs that have a frequency count greater than σ appended to the set M.

The two algorithms have two principal advantages: (i) the nature of the value of σ has very little (no?) effect on the algorithm's run time (as will be demonstrated later in Sub-section VI-B) and (ii) individual shapes can be considered in isolation hence the algorithm is well suited to parallelisation.

V. Movement Pattern Prediction

Once a set of frequently occurring MPs have been mined from a network $G = \{V, E\}$ we will wish to apply them to predict movement (traffic) in a previously unseen but related network G'. Note that the previously unseen network G' will be comprised solely of vertices. The pseudocode for the prediction mechanism is given in Algorithm 3. The algorithm takes as input a set of previously derived MPs M and a network $G' = \{V', E'\}$ where $E' = \emptyset$. Note that V is not necessarily equal to V'. The output is a set of predicted MPs I; on start up $I = \emptyset$. The algorithm commences by looping through all MPs in M. For each MP the "from" and "to" vertex attribute-value sets ($From_i$ and To_i) are extracted. The algorithm then loops through V' to check if $From_i$ is a subset of any $v_j \in V'$. If so the algorithm loops through V' again to check if To_i is a subset of any $v_k \in V'$. If so the edge attribute-value set is extracted from MP_i and used to form the tuple $\langle From_i, Edge_i, To_i\rangle$ which is added to the set i so far (the set of predicted MPs).

VI. Experiments and Evaluation

Experiments to determine the efficiency of the proposed ShaMP algorithms were conducted using a collection of artificially generated networks whose parameters could be controlled; the metric used was run time (seconds). Experiments concerning the accuracy of the prediction mechanism were conducted by applying the algorithm to a collection of large networks, extracted from the GB Cattle Tracking System (CTS) database, where the MPs were known. The evaluation metrics here were precision, recall and F1. The objectives of the evaluation were as follows:

Input:
M = Set of frequently occurring MPs $M = \{\langle MP_1, count_1\rangle, langleMP_2, count_2\rangle, \dots\}$ describing trafic a network G
G' = A previously unseen net work $G' = \{V', E'\}$ where $E' = \emptyset$
Output:
I = Set of predicted MPs in G', $I = \{\langle MP'_1, count'_1\rangle, langleMP'_2, count'_2\rangle, \dots\}$
Start:
for $MP_i \in M$ **do**
  $From_i$ = From attribute value set extracted from MP_i
  To_i = To attribute value set extracted from MP_i
  for $v_j \in V'$ **do**
    if $From_i \subseteq v_j$ **then**
      for $v_k \in V'$ **do**
        if $To_i \subseteq v_k$ *and* $v_j \neq v_k$ **then**
          $Edge_i$ = Edge attribute values set extracted from MP_i
          I = I $\cup \langle From_i, Edge_i, To_i\rangle$
        end
      end
    end
  end
end

Algorithm 3: Prediction Mechanism

1) To compare the operation of Shape approach versus a traditional Apriori based approach in term of support threshold and number of edges.
2) To determine the effect on the ShaMP algorithms of the size of the network under consideration in terms of the number of edges.
3) To compare the operation of ShaMP MPJ versus ShaMP Haddoop.
4) To determine the effectiveness of the prediction mechanism by comparing the set of predicted MPs with the set of known MPs.

A. Data Sets

Experiments were conducted using both artificial and real networks. The reason why artificially generated networks were required was that the number of edges and vertices could be controlled as well as the nature of the sets A_V and A_E. A purpose built network generator was used. For each generated artificial network the from and to vertices, for the specified number of edges, were selected at random ("self referencing" was not permitted). As a consequence the *degree* for each vertex was also generated randomly; the average number of inward or outward edges could be calculated using $|V|/|E|$. As in the case of the CTS networks (see below) the artificially generated networks were typically unconnected and featured some vertices with degree zero.

With respect to real networks, the CTS database associated the UK's cattle movement tracking system was used; this is managed by The UK's Department for Environment, Food and Rural Affairs (DEFRA). The database records the movement of all cattle between pairs of locations in GB. As such, these locations can be viewed as nodes, and the movement of cattle as edges between node pairs. The database was used to generate a collection of time stamped networks where for each network the vertices represented cattle holding areas and the edges occurrences of cattle movement (traffic). The database was preprocessed so that each record represented a group of animals moved, of the same type, breed and gender, from a given "from location" to a given "to location" on the same day. The attribute set A_V comprised: (i) holding area type and (ii) county name. While the set A_E comprised: (i) number of cattle moved, (ii) breed, (iii) gender, (iv) whether the animals moved are beef animals or not, and (v) whether the animals moved are dairy animals or not. The ShaMP algorithm was designed to operate with binary valued data (as in the case of traditional frequent item set mining). The values for the *numCattle* attribute were thus ranged into five sub ranges. The end result of the normalisation/discretisation exercise was an attribute value set comprising 391 individual attribute values.

The CTS database, preprocessed as described above, was then used to generate four networks covering the years 2003, 2004, 2005 and 2006 respectively. The number of vertices in each network was about $43,000$, while the number of edges was about $270,000$. Given the foregoing, and using Equation 1, the number of shapes that will need to be considered, in the context of the CTS networks, by the ShaMP algorithms, will thus be $|ShapeSet| = 2^2 - 1 \times 2^5 - 1 \times 2^2 - 1 = 3 \times 15 \times 3 = 279$.

B. Shape approach versus traditional Apriori approache

This section presents two sets of experiments conducted to compare the operation of the Shape based approach with a more tradition "Apriori" approach taken from the domain of Frequent Itemset Mining (FIM) [12]. At face value the MP Apriori based approach operates in a very similar manner to the FIM Apriori based approach in that it adopts a candidate generation, occurrence count and prune cycle. However, the distinction is that we are dealing with three part MPs ($\langle F, E, T\rangle$) and that none of these parts should be empty. Note that all the experiments were conducted using a single machine because the Apriori approach does not lend itself to parallelisation.

The first set of experiment considered in this section were concerned with efficiency using a range different support thresholds values (σ). In the context of FIM the lower the σ value the more frequent itemsets that will be found. Low σ values are seen as desirable because by finding many frequent itemsets there is less chance of missing anything significant, the same is true for MPs. In terms of FIM it is well established that efficiency decreases as the number of potential frequent itemsets increases (as the value of σ decreases). It was anticipated that this would also be true with

respect to the Apriori based MP mining. How this would effect the Shape approach was unclear, although it was conjectured that changing σ values would have little effect. Experiment were thus conducted using a range of σ values from 2.0 to 0.5 decreasing in steps of 0.5, and using the 2003 CTS movement networks described above ($|D| = 2,712,603$). The results are presented in Figure 4 where, for each graph, the x-axis gives the σ values and the y-axis runtimes in seconds. As expected, in the case of the Apriori approach, runtime increases exponentially as σ decreases. However, as conjectured, from the figure it can clearly be seen that $sigma$ has little effect on the Shape approach. It is interesting to note that it is not till σ drops below 1.0 that usage of the Shape approach becomes more advantageous than usage of the Apriori approach. However, we wish to identify as many relevant MPs as possible and to do this we would need to use very low σ values ($\sigma \leq 1.0$).

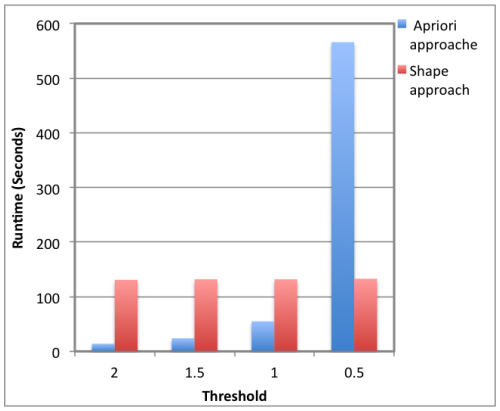

Fig. 4: Runtime (secs.) comparison between ShaMP and Apriori using a range of σ values.

In the second set of experiments the correlation between the two approaches and number of edges was tested using artificial data. In total five data sets were generated of increasing size: 100000, 300000, 500000, 1000000 and 2000000. The two approaches were tested on each of the data sets using $\sigma = 0.5$. The results are presented in Figure 5. From the figures it can clearly be seen that, as was to be expected, the run time increased as the number of records considered increased. However, the Shape approach is clearly more efficient than the Apriori approach. It can also be seen that the run time with respect to the Apriori approach increases dramatically as the number of edges increases, while the increase with respect to the Shape approach is much less dramatic.

C. Comparison of ShaMP MPJ and ShaMP Haddoop

This section presents the results obtained with respect to experiments conducted to compare the efficiency of operation of the proposed ShaMP MPJ and ShaMP Haddoop algorithms. For this purpose the four CTS networks and a sequence of five artificial networks were used.

The efficiency results with respect to the CTS network datasets, using ShaMP MPJ and ShaMP Haddoop, are presented in Figure 6. From the figure it can clearly be seen that the run time for ShaMP MPJ is much faster than for ShaMP Hadoop. The reason for this was because the nature of MPI was more beneficial where the massages passed between processes were not too large as in case of CTS network (number of edges less than 3 million). As the number of edges increased the runtime for ShaMP MPJ and ShaMP Hadoop would also increased. However, there will be a point where one of these algorithms can no longer process the number of edges to be considered. To determine where this point might be a sequence of five artificial data sets was generated that featured increasing numbers of edges ranging from $5,000,000$ to $25,000,000$ in steps of $5,000,000$. The number of vertices in the artificial networks was maintained at $|V| = 2,000,000$ in all cases. In addition, for the artificial data sets $|A_v| = 2$ and $|A_E| = 5$ were used because these were the values featured in the CTS datasets. For the experiments $\sigma = 1.0$ was used. Figure 7 presents the results obtained using the artificial data sets. From the figure it can be seen that there is no significant differences in operation between ShaMP MPJ and ShaMP Haddoop for the first two artificial data sets where the number of edges ≤ 10 million. However, as the number of

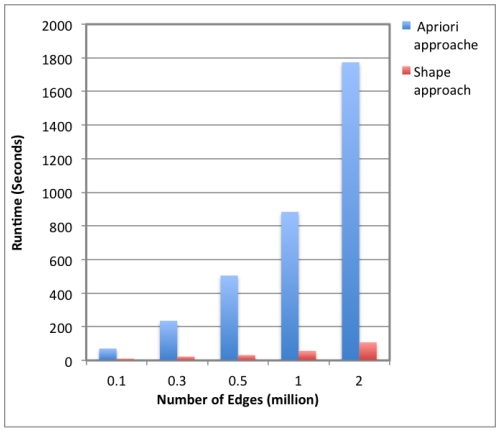

Fig. 5: Runtime (secs.) comparison between ShaMP and Apriori using a sequence of artificial networks.

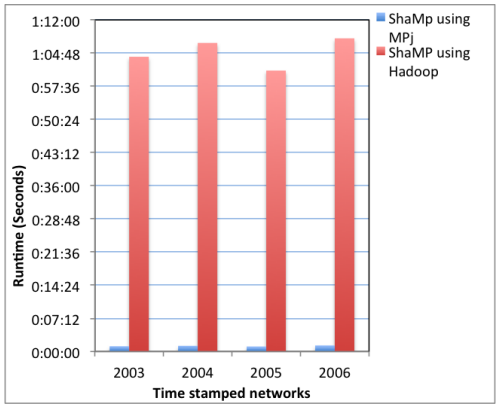

Fig. 6: Runtime comparison between ShaMP MPJ and ShaMP Hadoop using CTS networks.

edges is increased, the ShaMP MPJ algorithm was no longer be able to process the datasets as the extent of the message passing between machines became restrictive. On the other hand, the ShaMP Haddoop algorithm was able to process all five artificial networks. The reason for this relates to the Hadoop Distributed File System (HDFS) and the MapReduce process. Hadoop splits files into blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed.

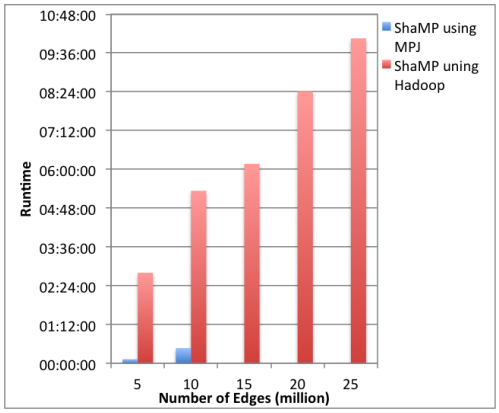

Fig. 7: Runtime comparison between ShaMP MPJ and ShaMP Hadoop using artificial nrtworks.

D. Accuracy of Prediction Mechanism

The accuracy of the proposed prediction mechanism was tested using the CTS network datasets where the edges (MPs) were known. The evaluation was conducted by applying the MPs generated with respect to one year to the vertices of all other years and determining the frequent MPs that resulted (using the same σ value). In each case the resulting set M' of predicted MPs was then compared with the set of known "ground truth" MPs M_T. As noted above, the performance measures used were Precision, Recall and the F1 measure [24]. Note that high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for precision and recall show that the predictor is performing well. The F1 measure combines both precision and recall and is thus the most significant measure.

The results are presented in Tables I, II, III and IV. Each table shows the application of MPs generated from a particular CTS data sets (2003, 2004, 2005 and 2006 respectively) to the remaining three CTS datasets. Concentrating on the F1 measure average values of 0.945, 0.951, 0.935 and 0.932 were generated, a overall average of 0.941, a very good result. The overall average values for precision and recall were 0.965 and 0.919 respectively, again a very good result.

VII. Conclusion

In this paper the authors have proposed: (i) the ShaMP algorithm (two variations using MPJ and Hadoop) for identifying Movement Pattens (MPs) in network data and (ii) a prediction mechanism for applying the identified (MPs) to previously unseen networks (where we wish to predict the edges/traffic). The MPs were defined in terms of a three parts tuple; From-Edge-To. The acronym FET was coined to describe such patterns. The three part MP concept has some similarities with traditional frequent itemsets except that the attributes that can appear in a particular part is limited, consequently the search space can be considerably reduced. A particular challenge was the size of the networks that to be analyse. The evaluation was conducted using artificial movement networks and movement networks extracted from the GB Cattle Tracking System (CTS) in operation in GB. The evaluation firstly indicated that, although the ShaMP MPJ approach seemed the most appropriate given its speed, the size of the networks to be considered meant that ShaMP Hadoop was in many cases more desirable. The evaluation secondly indicated that MPs could be effectively used to predict traffic (movement) in previously unseen networks. The main findings may thus be summarised as fallows: ShaMP MPJ can successfully identify MPs in small networks with reasonable computational efficiency, on the other hand ShaMP Hadoop is more suitable with respect to much larger networks;

TABLE I: Accuracy of proposed prediction mechanism using MPs generated from the 2003 CTS data set.

Network from which MPs were extracted	G' network = 2003		
	percision	**Recall**	**F1**
2004	0.978	0.967	0.973
2005	0.967	0.89	0.927
2006	0.961	0.929	0.945
average	0.961	0.929	0.945

TABLE II: Accuracy of proposed prediction mechanism using MPs generated from the 2004 CTS data set.

Network from which MPs were extracted	G' network = 2004		
	percision	**Recall**	**F1**
MPs from 2003	0.971	0.971	0.971
2005	0.979	0.895	0.935
2006	0.946	0.948	0.947
average	0.965	0.938	0.951

TABLE III: Accuracy of proposed prediction mechanism using MPs generated from the 2005 CTS data set.

Network from which MPs were extracted	G' network = 2005		
	percision	**Recall**	**F1**
2003	0.95	0.917	0.933
2004	0.962	0.918	0.939
2006	0.954	0.913	0.933
average	0.955	0.916	0.935

TABLE IV: Accuracy of proposed prediction mechanism using MPs generated from the 2006 CTS data set.

Network from which Ms were extracted	G'network = 2006		
	percision	**Recall**	**F1**
2003	0.957	0 .863	0.907
2004	0.968	0.889	0.927
2005	0.991	0.817	0.896
Average	0.979	0.892	0.932

the proposed prediction mechanism demonstrated that accurate traffic predictions could be made using the MP concept.

VIII. ACKNOWLEDGMENT

The first author would like to thank the Iraqi Ministry of Higher Education and Scientific Research and University of Al-Qadisiyah for partially funding this research.

REFERENCES

[1] N. Matsumura, D. E. Goldberg, and X. Llorà, "Mining directed social network from message board," in *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM, 2005, pp. 1092–1093.

[2] B. Chandrasekaran, "Survey of network traffic models," *Waschington University in St. Louis CSE*, vol. 567, 2009.

[3] S. Datta, K. Bhaduri, C. Giannella, R. Wolff, and H. Kargupta, "Distributed data mining in peer-to-peer networks," *Internet Computing, IEEE*, vol. 10, no. 4, pp. 18–26, 2006.

[4] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag, "Adaptive fastest path computation on a road network: a traffic mining approach," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 794–805.

[5] L. Tang and H. Liu, "Community detection and mining in social media," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 1–137, 2010.

[6] C. Kiss and M. Bichler, "Identification of influencers—measuring influence in customer networks," *Decision Support Systems*, vol. 46, no. 1, pp. 233–253, 2008.

[7] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[8] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface*. MIT press, 1999, vol. 1.

[9] A. Raorane and R. Kulkarni, "Data mining techniques: A source for consumer behavior analysis," *arXiv preprint arXiv:1109.1202*, 2011.

[10] J. Gudmundsson, P. Laube, and T. Wolle, "Movement patterns in spatio-temporal data," in *Encyclopedia of GIS*. Springer, 2008, pp. 726–732.

[11] W. M. Campbell, C. K. Dagli, and C. J. Weinstein, "Social network analysis with content and graphs," *Lincoln Laboratory Journal*, vol. 20, no. 1, 2013.

[12] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[13] C. A. Bliss, M. R. Frank, C. M. Danforth, and P. S. Dodds, "An evolutionary algorithm approach to link prediction in dynamic social networks," *Journal of Computational Science*, vol. 5, no. 5, pp. 750–764, 2014.

[14] M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks." in *SDM*, vol. 11. SIAM, 2011, pp. 47–58.

[15] F. Coenen, G. Goulbourne, and P. Leng, "Tree structures for mining association rules," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 25–51, 2004.

[16] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 1–12.

[17] F. Coenen and P. Leng, "Partitioning strategies for distributed association rule mining," *The Knowledge Engineering Review*, vol. 21, no. 01, pp. 25–47, 2006.

[18] N. Li, L. Zeng, Q. He, and Z. Shi, "Parallel implementation of apriori algorithm based on mapreduce," in *Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), 2012 13th ACIS International Conference on*. IEEE, 2012, pp. 236–241.

[19] Z. Farzanyar and N. Cercone, "Efficient mining of frequent itemsets in social network data based on mapreduce framework," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2013, pp. 1183–1188.

[20] A. Vishnu and K. Agarwal, "Large scale frequent pattern mining using mpi one-sided model," in *2015 IEEE International Conference on Cluster Computing*. IEEE, 2015, pp. 138–147.

[21] A. Javed and A. Khokhar, "Frequent pattern mining on message passing multiprocessor systems," *Distributed and Parallel Databases*, vol. 16, no. 3, pp. 321–334, 2004.

[22] J. Galloway and S. J. Simoff, "Network data mining: methods and techniques for discovering deep linkage between attributes," in *Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling-Volume 53*. Australian Computer Society, Inc., 2006, pp. 21–32.

[23] C. C. Aggarwal, "Applications of frequent pattern mining," in *Frequent Pattern Mining*. Springer, 2014, pp. 443–467.

[24] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel *et al.*, "Performance measures for information extraction," in *Proceedings of DARPA broadcast news workshop*, 1999, pp. 249–252.