MapReduce-Based Complex Big Data Analytics over Uncertain and Imprecise Social Networks

Peter Braun[1], Alfredo Cuzzocrea[2,3], Fan Jiang[1], Carson Kai-Sang Leung[1(✉)], and Adam G.M. Pazdor[1]

[1] University of Manitoba, Winnipeg, MB, Canada
kleung@cs.umanitoba.ca
[2] University of Trieste, Trieste, TS, Italy
alfredo.cuzzocrea@dia.units.it
[3] ICAR-CNR, Rende, CS, Italy

Abstract. With advances in technology, high volumes of valuable but complex data can be easily collected and generated from various sources in the current era of big data. A prime source of these complex big data is the social network, in which users are often linked by some interdependencies such as friendships and follower-followee relationships. These interdependencies can be uncertain and imprecise. Moreover, as the social network keeps growing, there are situations in which individual users or businesses want to find those popular (i.e., frequently followed) groups of users so that they can follow the same groups. In this paper, we present a complex big data analytic solution that uses the MapReduce model to mine uncertain and imprecise social networks for discovering groups of potentially popular users. Evaluation results show the efficiency and practicality of our solution in conducting complex big data analytics over uncertain and imprecise social networks.

1 Introduction and Related Work

With advancements in technology and the popularity of social networking sites (e.g., Facebook, Twitter [30,31]), high volumes of valuable but complex data [3,9,28] can be easily collected or generated from social networks [2,36]. In general, social networks are made of social entities (e.g., individuals, corporations, collective social units, or organizations) that are linked by some specific types of interdependencies (e.g., friendships, common interest, follower-followee relationships). A social entity is connected to another entity as friend, classmate, co-worker, team member, and/or business partner. For instance, Facebook users can create a personal profile, add other Facebook users as friends, exchange messages, and join common-interest user groups. The number of (mutual) friends may vary from one Facebook user to another. It is not uncommon for a user A to have hundreds or thousands of friends. Note that, although many of the Facebook users are linked to some other Facebook users via their mutual friendship (i.e., if a user A is a friend of another user B, then B is also a friend of A), there are situations in which such a relationship is not mutual. To handle these

L. Bellatreche and S. Chakravarthy (Eds.): DaWaK 2017, LNCS 10440, pp. 130–145, 2017.
DOI: 10.1007/978-3-319-64283-3_10

situations, Facebook added the functionality of "follow", which allows a user to subscribe or follow public postings of some other Facebook users without the need of adding them as friends. So, for any user C, if many of C's friends followed some individual users or groups of users, then C might also be interested in following the same individual users or groups of users. Furthermore, the "like" button allows users to express their appreciation of content such as status updates, comments, photos, and advertisements.

As another instance, Twitter users can read the tweets of other users by "following" them. Relationships between social entities are mostly defined by following (or subscribing) each other. Each user (social entity) can have multiple followers, and follows multiple users at the same time. The follow/subscribe relationship between follower and followee is not the same as the friendship relationship (in which each pair of users usually know each other before they setup the friendship relationship). In contrast, in the follow/subscribe relationship, a user D can follow another user E while E may not know D in person. In this paper, $D \rightarrow E$ denotes the follow/subscribe (i.e., "following") relationship that D is following E.

Big data analytics of these complex big social networks computationally facilitates social studies and human-social dynamics in these networks, as well as designs and uses information and communication technologies for dealing with social context. In this paper, we focus on a particular class of social networks called *imprecise and uncertain social networks*. Here, the main challenges due to the fact that edges are weighted by an *existential probability* [1,17,24,33].

With the growing number of users of these social networking sites, big data analytics has become very useful in order to extract useful and actionable knowledge from enormous data repositories (e.g., [4]), with a plethora of applications. In this methodological context, several data mining algorithms and techniques (e.g., [7,13,19,22,23]) have been proposed over the past two decades. Many of them (e.g., [18,25,29]) have been applied to mine social networks (e.g., discovery of special events (e.g., [11]), detection of communities (e.g., [27,34]), sub-graph mining (e.g., [35]), as well as discovery of popular friends (e.g., [14,25]), influential friends (e.g., [26]) and strong friends (e.g., [32])).

To discover "following" relationships among the aforementioned social networking sites, we developed a serial algorithm called FoP-miner [12] in DaWaK 2014 to mine interesting "following" patterns from social networks. To speed up the mining process, we extended the FoP-miner algorithm to get a parallel and concurrent algorithm called ParFoP-miner [21] for mining interesting "following" patterns in parallel. Moreover, in order to deal with massive numbers of these "following" relationships that are embedded in big data, we also extended the FoP-miner algorithm to get a MapReduce-based algorithm called BigFoP [20] in DaWaK 2015 for big data analytics of social networks and discovery of "following" patterns. These three algorithms work well when mining *precise* social network, in which the social analysts have precise information regarding the interdependencies among the social entities.

However, there are real-life situations in which this information is limited or unavailable. For instance, due to privacy preserving purposes, this information is not fully revealed. As such, social analysts may need to rely on their experience, expertise, or other sources to determine the likelihood of the existence of some of these interdependencies among the social entities within a complex social network with uncertainty and imprecision. In response, our *key contribution* of the current paper is our proposal of a new big data analytics and mining solution, which uses the *MapReduce* model [10] to discover *interesting popular patterns* consisting of social entities (or their social networking pages) that are frequently followed by social entities in a complex social network with *uncertain and imprecise* social data. Discovery of these patterns helps individual users find popular groups of social entities so that they can follow the same groups. Moreover, many businesses have used social network media to either (*i*) reach the right audience and turn them into new customers or (*ii*) build a closer relationship with existing customers. Hence, discovering those who follow collections of popular social networking pages about a business (i.e., discovering those who care more about the products or services provided by a business) helps the business identify its targeted or preferred customers.

The remainder of this paper is organized as follows. The next section provides some background on data science. Then, Sect. 3 describes our new data analytics solution, which uses the MapReduce model to discover interesting popular patterns from complex, big, uncertain and imprecise social networks in Sect. 4 observes and discusses evaluation results. Finally, Sect. 5 gives conclusions and future work.

2 Background: Data Science

Data science aims to develop systematic or quantitative processes to analyze and mine big data for continuous or iterative exploration, investigation, and understanding of past business performance so as to gain new insight and drive science or business planning. By applying *big data analytics and mining* (which incorporates various techniques from a broad range of fields such as cloud computing, data analytics, data mining, machine learning, mathematics, and statistics), data scientists can extract implicit, previously unknown, and potentially useful information from big data (e.g., big social network data).

In the past few years, researchers have used the high-level programming model MapReduce to process high volumes of big data by using parallel and distributed computing on large clusters or grids of nodes (i.e., commodity machines) or clouds, which consist of a master node and multiple worker nodes. As implied by its name, MapReduce involves two key functions: (*i*) the map function and (*ii*) the reduce function. Specifically, the input data are read, divided into several partitions (sub-problems), and assigned to different processors. Each processor executes the *map function* on each partition (sub-problem). The map function takes a pair of $\langle key_1, value_1\rangle$ and returns a list of $\langle key_2, value_2\rangle$ pairs as an intermediate result, where (*i*) key_1 and key_2 are keys in the same or different domains and (*ii*) $value_1$ and $value_2$ are the corresponding values in some domains. Afterward, these pairs are shuffled and sorted. Each processor then executes the *reduce function* on (*i*) a single key key_2 from this intermediate result $\langle key_2,$ list of $value_2\rangle$ together with (*ii*) the list of all values that appear with this key in the intermediate result. The reduce function "reduces"—by combining, aggregating, summarizing, filtering, or transforming—the list of values associated with a given key key_2 (for all k keys) and returns a single (aggregated or summarized) value $value_3$, where (*i*) key_2 is a key in some domains and (*ii*) $value_2$ and $value_3$ are the corresponding values in some domains. An advantage of using the MapReduce model is that users only need to focus on (and specify) these "map" and "reduce" functions—without worrying about implementation details for (*i*) partitioning the input data, (*ii*) scheduling and executing the program across multiple machines, (*iii*) handling machine failures, or (*iv*) managing inter-machine communication. The construction of an inverted index as well as the word counting of a document for data processing [10] are a few examples of MapReduce applications.

3 Mining Complex Big Data in Uncertain and Imprecise Social Networks

Now, let us present our new big data analytics and mining solution—called **BigUISN**—for mining **big** **u**ncertain and **i**mprecise **s**ocial **n**etworks for interesting popular patterns using the MapReduce model.

3.1 Interdependencies Between Followers and Followees in Complex Big Social Networks

Social entities (i.e., users) in social networking sites like Twitter are linked by *"following" relationships* such as $A \rightarrow B$ indicating that a user A (i.e., *follower*) follows another user B (i.e., *followee*). Then, given a social network in which each social entity is *following* some other social entities, such a social network can be represented as a graph $G = (V, E)$ where (*i*) V is a set of vertices (i.e., social entities) and (*ii*) E is a set of weighted directional edges connecting some of these vertices (i.e., "following" relationships). See Example 1.

Example 1. For illustrative purpose, let us consider a small portion of a complex big social network as shown in Fig. 1. It can be represented by $G = (V, E)$, where (*i*) $V = \{$*Alain, Benoit, Charlot, Denis, Emile, Frederic*$\}$ and (*ii*) $E = \{\langle$*Alain, B*$\rangle$:0.9, $\langle$*Alain, E*$\rangle$:0.9, $\langle$*Benoit, A*$\rangle$:1, $\langle$*Benoit, C*$\rangle$:1, $\langle$*Benoit, E*$\rangle$:1, $\langle$*Charlot, A*$\rangle$:0.7, $\langle$*Charlot, E*$\rangle$:0.7, $\langle$*Denis, B*$\rangle$:1, $\langle$*Denis, C*$\rangle$:1, $\langle$*Denis, E*$\rangle$:1, $\langle$*Emile, A*$\rangle$:0.8, $\langle$*Emile, B*$\rangle$:0.8, $\langle$*Emile, C*$\rangle$:0.8, $\langle$*Emile, D*$\rangle$:0.8, $\langle$*Frederic, A*$\rangle$:1, $\langle$*Frederic, B*$\rangle$:1, $\langle$*Frederic, C*$\rangle$:1, $\langle$*Frederic, E*$\rangle$:1$\}$. □

In contrast to the mutual friendship relationships, the "following" relationships are different in that the latter are *directional*. For instance in Example 1, *Benoit* is following *Charlot*, but *Charlot* is not following *Benoit*. This property increases the complexity of the problem because of the following reasons.

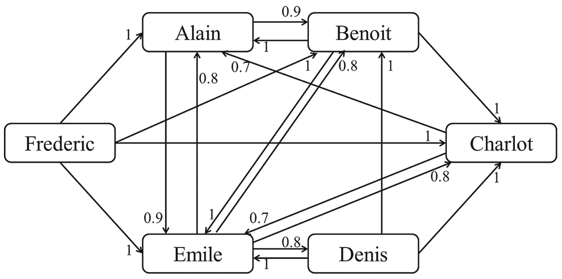

Fig. 1. A sample social network with uncertain and imprecise information about $|V| = 6$ users.

The group of users followed by *Benoit* (e.g., *Benoit* $\rightarrow$ {*Alain, Charlot, Emile*}) may not be same group of users as those who are following *Benoit* (e.g., {*Alain, Denis, Emile, Frederic*} $\rightarrow$ *Benoit*). Hence, we need to store directional edges (e.g., ⟨*Alain, Benoit*⟩, ⟨*Benoit, Alain*⟩) instead of undirected edges (e.g., {*Alain, Benoit*} indicating that *Alain* and *Benoit* are mutual friends). Given $|V|$ social entities, there are potentially $|V|(|V|-1)$ directional edges for "following" relationships (cf. potentially $\frac{|V|(|V|-1)}{2}$ undirected edges for mutual friendship relationships). In addition to an increase in storage space, the computation time also increases because we need to check both directions to get relationships between pairs of users (e.g., cannot determine whether or not *Charlot* $\rightarrow$ *Benoit* if we only know *Benoit* $\rightarrow$ *Charlot*).

Moreover, as we focus on imprecise and uncertain social networks in this paper, each edge (u, v) in the network is associated with an *existential probability* to indicate the likelihood for social entity u to "follow" another social entity v. For instance, due to the privacy setting of some social entities, analysts may suspect—but cannot guarantee—that a user *Alain* is likely to follow user *Benoit*. Such suspicion can be expressed in terms of existential probability. In Example 1, an edge ⟨Alain, Benoit⟩ is associated with an existential probability value in the range of $(0, 1]$—namely, 0.9, which expressed that *Alain* has a 90% chance of following *Benoit*. In other words, *Alain* has a 10% chance of *not* following *Benoit*.

Besides the privacy setting, another source of uncertainty or imprecision is the ambiguity in name. For instance, a user may know a friend by nick name or common name (say, *Johnny*), which may not be his official name or the name used in the social networking sites (say, *Jean*). In this situation, the user may be uncertain about which social entities to follow? Should that user follow *Johnny*, *John*, *Jean*, *Juan*, *Hans*, or *Ivano*? Once again, such uncertainty can be expressed in terms of existential probability with value in the range of $(0, 1]$.

3.2 Discovery of Popular Followees

With the explosive growth of the number of users in social networking sites (e.g., Twitter), the number of "following" relationships between followers and followees in complex social network are also growing. One of the important research problems with regard to this high volume of data is to discover interesting "following" patterns.

A popular pattern is a pattern representing the linkages when a significant number of users (i.e., followers) following the same combination/group of users (i.e., followees). For example, users who follow the twitter feed or tweets of UBC also follow the tweets of its President. If there are large numbers of users who follow the tweets of both UBC and its university President together, we can define this combination ({UBC, President Santa Ono}) of followees as an interesting popular pattern (i.e., a frequently followed group).

To discover interesting popular patterns (i.e., collections of social network pages that are frequently followed by users), we propose a multi-step data science solution called **BigUISN** for mining complex big data from uncertain and imprecise social networks by using sets of map and reduce functions.

3.3 The First Set of MapReduce Functions in BigUISN

In high-level abstract terms, **BigUISN** first applies a map function to each edge as follows:

$$map_1\colon \langle edgeID, followingRel\rangle \rightarrow \langle follower\ F, indFollowee\ f, Pr(F,f)\rangle \qquad (1)$$

where: (i) *edgeID* is the identifier of the edge; (ii) *followingRel* is the "following" relationship captured by the edge with ID *edgeID*; (iii) *follower* is the follower, denoted hereinafter as F; (iv) *indFollowee* is the individual followee, denoted hereinafter as f; and (v) $Pr(F,f)$ is the existential probability of F following f. Looking in more implementation details, the master node reads edges modeled like that, and divides big social network data in partitions. Specifically, the map_1 function is detailed by Algorithm 1.

Algorithm 1. map_1

```
1: Input: social network $G = (V, E)$
2: Output: list of objects $\langle F, f, Pr(F,f)\rangle$
3: for each edge $e = \langle F, f\rangle \in E$ do
4:     emit $\langle F, f, Pr(F,f)\rangle$;
```

Map function map_1 is applied to each edge $e = \langle F, f\rangle \in E$—with an existential probability $0 < Pr(F,f) \leq 1$—in the social network $G = (V, E)$, and provides as result in a list of $\langle F, f, Pr(F,f)\rangle$ capturing all potential "following" relationships (between followers and followees) in the social network. See Example 2.

Example 2. After applying the map_1 function to the social network data in Example 1, our **BigUISN** returns the following list: $\mathcal{L} = \{\langle Alain, B, 0.9\rangle$, $\langle Alain, E, 0.9\rangle$, $\langle Benoit, A, 1\rangle$, $\langle Benoit, C, 1\rangle$, $\langle Benoit, E, 1\rangle$, $\langle Charlot, A, 0.7\rangle$,

$\langle Charlot, E, 0.7\rangle$, $\langle Denis, B, 1\rangle$, $\langle Denis, C, 1\rangle$, $\langle Denis, E, 1\rangle$, $\langle Emile, A, 0.8\rangle$, $\langle Emile, B, 0.8\rangle$, $\langle Emile, C, 0.8\rangle$, $\langle Emile, D, 0.8\rangle$, $\langle Frederic, A, 1\rangle$, $\langle Frederic, B, 1\rangle$, $\langle Frederic, C, 1\rangle$, $\langle Frederic, E, 1\rangle\}$. □

Hereafter, **BigUISN** applies a reduce function to group and compute the expected number of followers for each followee, as well as to list these followers for each followee. This function is named as $reduce_1$, and its general form for $reduce_{k\geq1}$ reads as follows:

$$\begin{aligned} reduce_{k\geq1}\colon & \langle P_k, \text{list of } \langle F, P_k, Pr(F, P_k)\rangle\rangle \\ & \rightarrow \text{list of } \langle P_k, \text{expCnt}[P_k], \text{list}[P_k]\rangle \end{aligned} \quad (2)$$

For $k = 1$, followee group P_k is an individual social entity f returned by map_1. More specifically, $\langle F, f, Pr(F, f)\rangle$-tuplets from the map_1 function are shuffled and sorted (where $P_{k=1} = f$). Each processor then executes the reduce function on the shuffled and sorted pairs to compute the expected number of followers and list them for each followee. Here, the expected number of followers is computed as a product of the related existential probabilities. To speed up this big social network data mining process, **BigUISN** also allows users to specify the *interestingness* of groups of social entities by a frequency threshold τ. Here, the users can indicate the minimum number of followers for a group of followees so that the group can be considered interesting. By incorporating this user preference, **BigUISN** returns (i) a list of followers only for those popular followees (i.e., followees who are frequently followed by at least the minimum number of followers) and (ii) the expected number of each followee. The $reduce_{k\geq1}$ function is detailed by Algorithm 2.

Algorithm 2. $reduce_{k\geq1}$

```
1: Input: followee group $P_k$; list of $\langle F, P_k, Pr(F, P_k)\rangle$; min freq. threshold $\tau$
2: Output: list of $\langle$interesting group $P_k$ of $k$ followees, expCnt$[P_k]$, list$[P_k]\rangle$
3: for each followee group $P_k \in \langle \_, P_k, \_\rangle$ emitted by $map_k$ do
4:     set expCnt$[P_k] \leftarrow 0$; list$[P_k] \leftarrow \oslash$;
5:     for each follower $F \in \langle F, P_k, Pr(F, P_k)\rangle$ emitted by $map_k$ do
6:         expCnt$[P_k] \leftarrow$ expCnt$[P_k] + Pr(F, P_k)$; list$[P_k] \leftarrow$ list$[P_k] \cup F$;
7:     if (expCnt$[P_k] \geq \tau$) then
8:         emit $\langle P_k$, expCnt$[P_k]$, list$[P_k]\rangle$;
```

Reduce function $reduce_1$ results in (i) a list of followers and (ii) its expected number of each *interesting/popular* individual followee f. See Example 3.

Example 3. Let us continue with Example 2. **BigUISN** applies the $reduce_1$ function with user-specified minimum frequency threshold $\tau = 2$ followers and returns the following list: $\mathcal{L} = \{\langle A, 3.5, \{Benoit, Charlot, Emile, Frederic\}\rangle$, $\langle B, 3.7, \{Alain, Denis, Emile, Frederic\}\rangle$, $\langle C, 3.8, \{Benoit, Denis, Emile, Frederic\}\rangle$, $\langle E, 4.6, \{Alain, Benoit, Charlot, Denis, Frederic\}\rangle\}$. Note that our

BigUISN does not return the lists for followees D or F because their corresponding counters were low (D and F were expected to be followed by only 0.8 and 0 followers, respectively).

To recap, after applying the first set of map_1 and $reduce_1$ functions, our **BigUISN** has so far discovered four interesting popular patterns—in the form of *individual* frequently followed social entities—namely: $\{\{A\}, \{B\}, \{C\}$ and $\{E\}\}$, who are expected to be followed by 3.5, 3.7, 3.8 and 4.6 followers, respectively. In other words, each of these four individual followees is followed by at least $\tau = 2$ followers. □

3.4 The Second Set of MapReduce Functions in BigUISN

After applying the first set of MapReduce functions, **BigUISN** then applies a next set of map and reduce functions to mine interesting popular patterns in the form of *pairs* of frequently followed social entities based on the results from the first set of map_1 and $reduce_1$ functions. For instance, knowing that D and E are unpopular individual followees, it is guaranteed that any pairs containing followee D or E is also unpopular. By making use of this knowledge, the search space for mining interesting popular patterns can then be pruned effectively. In other words, the general form for $map_{k\geq 2}$ reads as follows:

$$map_{k\geq 2}\text{: list of}\langle P_{k-1}, \text{expCnt}[P_{k-1}], \text{list}[P_{k-1}]\rangle \rightarrow \text{list of } \langle F, P_k, Pr(F, P_k)\rangle \quad (3)$$

where $P_k = P_{k-1} \cup \{f\}$. For $k = 2$, followee group P_{k-1} is an individual followee p and followee group P_k is a followee pair $\{p, f\}$. More specifically, the map_2 function returns objects of kind $\langle F, \{p, f\}, Pr(F, \{p, f\})\rangle$ for every follower F in the follower list of each popular/interesting individual followee p. The map_2 function is detailed by Algorithm 3.

Algorithm 3. $map_{k\geq 2}$

```
1: Input: list of ⟨$P_{k-1}$, expCnt[$P_{k-1}$], list[$P_{k-1}$]⟩
2: Output: list of ⟨$F, P_{k-1} \cup \{f\}, Pr(F, P_{k-1} \cup \{f\})$⟩
3: for each interesting followee $P_{k-1} \in \langle P_{k-1}, \_, \text{list}[P_{k-1}]\rangle$ emitted by $reduce_{k-1}$ do
4:     for each follower $F \in \text{list}[P_{k-1}]$ do
5:         for each $\langle F, f\rangle \in E$ do
6:             if (isRelevant($f, P_{k-1}$)) then
7:                 emit $\langle F, P_{k-1} \cup \{f\}, Pr(F, P_{k-1} \cup \{f\})\rangle$;
```

Note that $isRelevant(f, P_{k-1})$ is a Boolean function checking the relevance (e.g., consistence to the mining order) of followee f with respect to P_{k-1}. Map function map_2 results in lists of $\langle F, P_k, Pr(F, P_k)\rangle$ where $P_k = P_{k-1} \cup \{f\}$. See Example 4.

Example 4. Continue with Example 3. Recall that the first set of map_1 and $reduce_1$ functions returns four popular followees A, B, C and E. So, for popular followee A (followed by four followers: $\{Benoit, Charlot, Emile, Frederic\}$), the map_2 function emits all *relevant* followees of these four followers, which are defined as follows: $\mathcal{L} = \{\langle Benoit, \{A, C\}, 1\rangle$, $\langle Benoit, \{A, E\}, 1\rangle$, $\langle Charlot, \{A, E\}, 0.49\rangle$, $\langle Emile, \{A, B\}, 0.64\rangle$, $\langle Emile, \{A, C\}, 0.64\rangle$, $\langle Frederic, \{A, B\}, 1\rangle$, $\langle Frederic, \{A, C\}, 1\rangle$, $\langle Frederic, \{A, E\}, 1\rangle\}$.

Note that: (i) followees of *Alain* are not emitted (because it is not meaningful for *Alain* to follow himself); (ii) followees of *Denis* are not emitted (because *Denis* does not follow A); (iii) four relationships in the form $\langle \_, A, \_\rangle$ (e.g., $\langle Benoit, A, 1\rangle$) are irrelevant with respect to $p = A$ (because we already knew these four followers are following *single individual followee* A when we started this map_2 function and we aimed to find followers who follow *pairs of followees*); (iv) $\langle Emile, \{A, D\}, 0.64\rangle$ is also irrelevant (because followee D is unpopular).

Similarly, for popular followee B (followed by four followers: {*Alain, Denis, Emile, Frederic*}), the map_2 function emits all *relevant* followee of these four followers: $\{\langle Alain, \{B, E\}, 0.81\rangle$, $\langle Denis, \{B, C\}, 1\rangle$, $\langle Denis, \{B, E\}, 1\rangle$, $\langle Emile, \{B, C\}, 0.64\rangle$, $\langle Frederic, \{B, C\}, 1\rangle$, $\langle Frederic, \{B, E\}, 1\rangle\}$. Note that: ($i$) followees of *Benoit* are not emitted (because it is not meaningful for *Benoit* to follow himself); (ii) followees of *Charlot* are not emitted (because *Charlot* does not follow B); (iii) four relationships in the form $\langle \_, B, \_\rangle$ (e.g., $\langle Denis, B, 1\rangle$) are irrelevant with respect to $p = B$ (because we already knew these four followers are following *single individual followee* B when we started this map_2 function and we aimed to find followers who follow *pairs of followees*); (iv) $\langle Emile, \{B, D\}, 0.64\rangle$ is also irrelevant (because followee D is unpopular); (v) relationships in the form $\langle \_, \{A, B\}, \_\rangle$ (e.g., $\langle Emile, \{A, B\}, 0.64\rangle$, $\langle Frederic, \{A, B\}, 1\rangle$) are irrelevant with respect to $p = B$ (because these relationships are already processed by the map_2 function).

Then, for popular followee C (followed by four followers: {*Benoit, Denis, Emile, Frederic*}), the map_2 function emits all *relevant* followee of these four followers: $\{\langle Benoit, \{C, E\}, 1\rangle$, $\langle Denis, \{C, E\}, 1\rangle$, $\langle Frederic, \{C, E\}, 1\rangle\}$.

Finally, for popular followee E (followed by five followers: {*Alain, Benoit, Charlot, Denis, Frederic*}), the map_2 function does not emit any followee because there is no *relevant* followee for these five followers. □

In similarity to the $reduce_1$ function, $reduce_2$—which is also a specialization of Eq. (2)—shuffles and sorts objects of kind $\langle F, \{p, f\}, Pr(F, \{p, f\})\rangle$ to find and compute followers for each followee pair $P_2 = \{p, f\}$, as detailed by Algorithm 2. Reduce function $reduce_2$ results in (i) a list P_k of followers and (ii) its expected number of each *interesting/popular* followee pair $P_2 = \{p, f\}$. See Example 5.

Example 5. Let us continue with Example 4. Our **BigUISN** applies the $reduce_2$ function with user-specified minimum frequency threshold $\tau = 2$ followers and returns the following list: $\mathcal{L} = \{\langle\{A, C\}, 2.64, \{Benoit, Emile, Frederic\}\rangle$, $\langle\{A, E\}, 2.49, \{Benoit, Charlot, Frederic\}\rangle$, $\langle\{B, C\}, 2.64, \{Denis, Emile, Frederic\}\rangle$, $\langle\{B, E\}, 2.81, \{Alain, Denis, Frederic\}\rangle$, $\langle\{C, E\}, 3, \{Benoit, Denis, Frederic\}\rangle\}$.

Hence, after applying this second set of map_2 and $reduce_2$ functions, our **BigUISN** algorithm discovered five interesting "following" patterns—in the form of pairs of frequently followed social entities—namely: $\{\{A, C\}$, $\{A, E\}$,

$\{B, C\}$, $\{B, E\}$, $\{C, E\}\}$, who are expected to be followed by 2.64, 2.49, 2.64, 2.81 and 3 followers, respectively. Each of these five followee pairs is thus followed by at least $\tau = 2$ followers. □

3.5 Beyond the Second Set of MapReduce Functions in BigUISN

So far, **BigUISN** has found interesting popular patterns in the form of (i) *individual* frequently followed social entities as well as (ii) *pairs* of frequently followed social entities. **BigUISN** then applies *similar* sets of map and reduce functions to find triplets, quadruplets, quintuplets and higher (i.e., k-tuplets for $k \geq 3$) of frequently followed social entities. See Example 6.

Example 6. Let us continue with Example 5. For popular followee group $\{A, B\}$ (followed by two followers: $\{Emile, Frederic\}$), the map_3 function emits the following three *relevant* followees: $\{\langle Emile, \{A, B, C\}, 0.512\rangle$, $\langle Frederic, \{A, B, C\}, 1\rangle$, $\langle Frederic, \{A, B, E\}, 1\rangle\}$.

For popular followee group $\{A, C\}$ (followed by three followers: $\{$*Benoit, Emile, Frederic*$\}$), the map_3 function emits the following two *relevant* followees: $\{\langle Benoit, \{A, C, E\}, 1\rangle$, $\langle Frederic, \{A, C, E\}, 1\rangle\}$.

In a similar fashion, the map_3 function emits the following two *relevant* followees: $\{\langle Denis, \{B, C, E\}, 1\rangle$, $\langle Frederic, \{B, C, E\}, 1\rangle\}$ for popular followee group $\{B, C\}$ (followed by three followers: $\{Denis, Emile, Frederic\}$).

Hereafter, by applying the $reduce_3$ function, **BigUISN** discovers the following two interesting "following" patterns $\varphi_1 = \{A, C, E\}$ and $\varphi_2 = \{B, C, E\}$, with their associated lists and number of followees, which are defined as follows: $\mathcal{L} = \{\langle\{A, C, E\}, 2, \{Benoit, Frederic\}\rangle$, $\langle\{B, C, E\}, 2, \{Denis, Frederic\}\rangle\}$.

Based on the results returned by the $reduce_3$ function, **BigUISN** applies map_4 but returns nothing because there is no relevant quadruplet of frequently followed social entities. This completes the mining process for interesting "following" patterns from our illustrative example social network. Note that key concepts and steps illustrated in this example are applicable to any uncertain and imprecise social network. □

4 Evaluation, Observations, and Discussion

Our **BigUISN** takes advantages of the MapReduce model when discovering popular patterns over uncertain and imprecise social networks. The input complex social data are divided into several partitions (sub-problems) and assigned to different processors. Each processor executes the map_k and $reduce_k$ functions (for $k \geq 1$). On the surface, one might worry that lots of communications or exchanges of information are required among processors. Fortunately, due to the divide-and-conquer nature of our big social network data analytics solution of discovering popular patterns, once the original big social network is partitioned and assigned to each processor (e.g., one processor is assigned the followers of A, another is assigned the followers of B, a third one is assigned the followers of C), each processor handles the assigned data without any reliance on the results from other processors. As observed from the above examples, the processor assigned for the followers of a popular followee can apply the subsequent sets of map and reduce functions on data emitted by that processor. For example, a processor applies map_1 and $reduce_1$ to find popular followee A. That processor can then apply map_2 on the data emitted by $reduce_1$ from that processor to find popular followee group $\{A, B\}$ (i.e., group containing A). Similarly, the processor applies map_3 on the data emitted by $reduce_2$ from the same processor to find subsequent popular followee group $\{A, B, C\}$. Without the need of extra communications and exchanges of data among processors, our **BigUISN** discovers all interesting popular patterns efficiently.

Moreover, if a partition of the complex big social network is too big to be handled by a single processor, our **BigUISN** furthers sub-divide that partition so that the resulting sub-partitions can be handled by each of the multiple processors. Furthermore, due to the divide-and-conquer nature of our big social network data analytics solution of discovering popular patterns, the amount of data input for the map_k and $reduce_k$ functions monotonically decreases as the size of the popular group of k followees increases. Our **BigUISN** discovers all interesting popular patterns in a space effective manner.

In terms of functionality, existing algorithms like FoP-miner [12], ParFoP-miner [21] and the BigFoP algorithm [20] mine popular patterns from precise social networks. In contrast, our current **BigUISN** algorithm is capable of handling both *precise* social networks as well as *uncertain and imprecise* social networks. Note that the former can be considered as a special case of the latter when $Pr(F, f) = 1$ for all existing edges $(F, f) \in E$ in the complex big social network represented as a social graph $G = (V, E)$. In other words, the former three algorithms can handle social graphs with every edge (F, f) having a weight (i.e., existential probability) of 1, whereas the latter (i.e., **BigUISN**) is more flexible in the sense that it can handle social graphs with any edge (F, f) of different weight (i.e., $0 < Pr(F, f) \leq 1$).

In terms of accuracy, when $Pr(F, f) = 1$ for all existing edges $(F, f) \in E$ in the complex big social network represented as a social graph $G = (V, E)$, all four algorithms—namely, FoP-miner [12], ParFoP-miner [21], the BigFoP algorithm [20], and our current **BigUISN** algorithm—gives the same results. In other words, all four algorithms return the same sets of popular (i.e., frequently followed) groups of users.

As another quality measure, we also compared the runtime performance of our **BigUISN** with related works (e.g., FoP-miner [12], ParFoP-miner [21], and the BigFoP algorithm [20]). Note that FoP-miner is a serial algorithm that discovers "following" patterns from precise social networks, ParFoP-miner is a parallel algorithm that discovers "following" patterns from precise social networks, and BigFoP is a MapReduce-based algorithm that discovers "following" patterns from precise social networks. In contrast, our **BigUISN** uses the MapReduce model for the discovery of "following" patterns from uncertain and imprecise

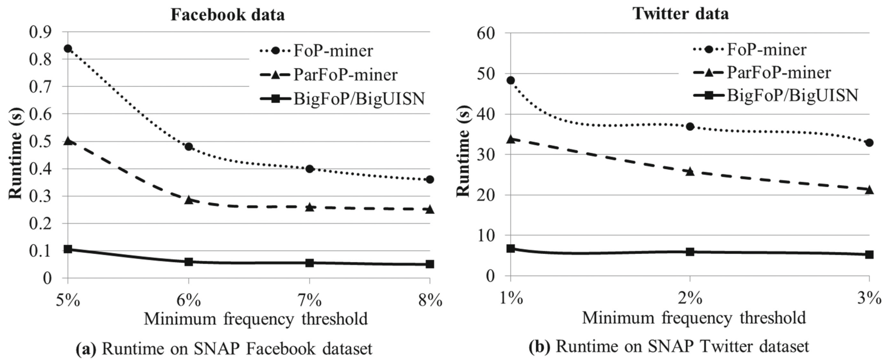

Fig. 2. Experimental results of **BigUISN** on social network data sets.

social networks. We used real-life social network data sets: *The Stanford Network Analysis Project* (SNAP) ego-Facebook data set and ego-Twitter data set[1]. The SNAP Facebook data set contains 4,039 social entities and 88,234 connections ("following" relationships) between these social entities. The SNAP Twitter data set contains 81,306 social entities and 1,768,149 connections between these social entities. All experiments were run using either (*i*) a single machine with an *Intel Core i7 4-core processor* (1.73 GHz) and 8 GB of main memory running a *64-bit Windows 7 operating system*, or (*ii*) the *Amazon Elastic Compute Cloud* (EC2) cluster—specifically, *11 High-Memory Extra Large (m2.xlarge) computing nodes*[2]. We implemented both the existing algorithms and our proposed **BigUISN** in the Java programming language. The stock version of *Apache Hadoop 2.6.5* was used. The results shown in Fig. 2, in which the x-axis shows the user-specified minimum frequency threshold (in percentage of the number of social entities) expressing the interestingness of the mined patterns, are based on the average of 10 runs. Runtime includes CPU and I/Os in the mining process of interesting "following" patterns. In particular, Fig. 2(a) shows that **BigUISN** provided a speedup of about 8 times when compared with FoP-miner, as well as a speedup of about 5 times when compared with ParFoP-miner, in mining the SNAP Facebook data set. Higher speedup is expected when using more processors. It is interesting to note that both BigFoP and **BigUISN** took almost the same runtime because the former can be considered as a special case of the latter where $Pr(F, f) = 1$ for every edge (F, f) in the complex big social networks. Figure 2(b) shows a similar result for the SNAP Twitter data set. Furthermore, our **BigUISN** is shown to be scalable with respect to the number of social entities in the big social network. In addition, we experimented with various existential probability values associated with edges. The results show that the runtime was stable regardless of the probability values.

[1] http://snap.stanford.edu/data/.
[2] http://aws.amazon.com/ec2/.

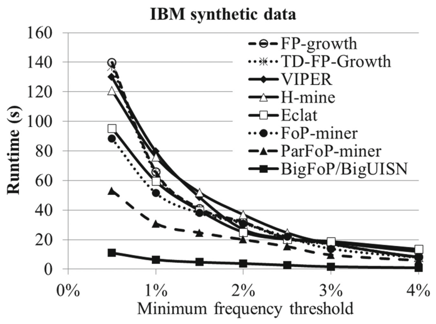

Fig. 3. Experimental results of **BigUISN** on an IBM synthetic data set.

We also experimented with other data sets. For instance, we compared **BigUISN** with existing frequent pattern mining algorithms. The results shown in Fig. 3 reveal the benefits of using **BigUISN**.

As ongoing work, we are conducting more experiments, including an in-depth study on the quality of discovered popular patterns.

As we can see from overall our experimental campaign, **BigUISN** not only ensures effectiveness and quality of results, but also performance. Indeed, as highlighted by many recent studies (e.g., [16]), performance is extremely important when processing social networks, especially those exposing big data repositories as underlying data layer.

5 Conclusions and Future Work

In this paper, we proposed a big data analytics and mining algorithm—called *BigUISN*—for discovering interesting popular patterns from big uncertain and imprecise social networks. **BigUISN** helps social network users to discover groups of frequently followed followees from complex big social networks with uncertain and imprecise social data by using the MapReduce model. By applying **BigUISN**, social network users (e.g., newcomers) could find popular groups of followees and follow them. Similarly, a business could find popular groups of followed products and services and incorporate customers' feedback on these products and services. Experimental results show the effectiveness of **BigUISN** as a MapReduce-based solution in this complex big data analytics task of conducting big social data analytics over uncertain and imprecise social networks for interesting popular patterns.

Future work is oriented towards enriching our framework with some innovative features, as to deal with emerging big data trends. Among these, we recall: (*i*) embedding data compression paradigms (e.g., [8]) as to improve efficiency; (*ii*) embedding data partition/fragmentation paradigms (e.g., [5,6]) as to improve distribution; (*iii*) exploring techniques to reduce the number of map-reduce functions used during the mining process; and (*iv*) exploiting alternative big data science frameworks such as the Spark framework (e.g., [15]).

Acknowledgement. This project is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) and University of Manitoba.

References

1. Balsa, E., Troncoso, C., Diaz, C.: A metric to evaluate interaction obfuscation in online social networks. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. **20**(6), 877–892 (2012)
2. Bohlouli, M., Dalter, J., Dornhöfer, M., Zenkert, J., Fathi, M.: Knowledge discovery from social media using big data-provided sentiment analysis (SoMABiT). J. Inf. Sci. **41**(6), 779–798 (2015)
3. Chen, C.L.P., Zhang, C.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. **275**, 314–347 (2014)
4. Cuzzocrea, A., Bellatreche, L., Song, I.-Y.: Data warehousing and OLAP over big data: current challenges and future research directions. In: ACM DOLAP 2013, pp. 67–70 (2013)
5. Cuzzocrea, A., Darmont, J., Mahboubi, H.: Fragmenting very large XML data warehouses via k-means clustering algorithm. Int. J. Bus. Intell. Data Min. **4**(3/4), 301–328 (2009)
6. Cuzzocrea, A., Furfaro, F., Saccà, D.: Hand-OLAP: a system for delivering OLAP services on handheld devices. In: ISADS 2003, pp. 80–87 (2003)
7. Cuzzocrea, A., Leung, C.K.-S., MacKinnon, R.K.: Mining constrained frequent itemsets from distributed uncertain data. Future Gener. Comput. Syst. **37**, 117–126 (2014)
8. Cuzzocrea, A., Saccà, D., Serafino, P.: A hierarchy-driven compression technique for advanced OLAP visualization of multidimensional data cubes. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 106–119. Springer, Heidelberg (2006). doi:10.1007/11823728_11
9. Cuzzocrea, A., Saccà, D., Ullman, J.D.: Big data: a research agenda. In: IDEAS 2013, pp. 198–203 (2013)
10. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
11. Dhahri, N., Trabelsi, C., Ben Yahia, S.: RssE-Miner: a new approach for efficient events mining from social media RSS feeds. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 253–264. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32584-7_21
12. Jiang, F., Leung, C.K.-S.: Mining interesting "following" patterns from social networks. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 308–319. Springer, Cham (2014). doi:10.1007/978-3-319-10160-6_28
13. Jiang, F., Leung, C.K.-S.: Stream mining of frequent patterns from delayed batches of uncertain data. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 209–221. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40131-2_18
14. Jiang, F., Leung, C.K.-S., Liu, D., Peddle, A.M.: Discovery of really popular friends from social networks. In: IEEE BDCloud 2014, pp. 342–349 (2014)
15. Jiang, F., Leung, C.K.-S., Sarumi, O.A., Zhang, C.Y.: Mining sequential patterns from uncertain big DNA data in the Spark framework. In: IEEE BIBM 2016, pp. 874–881 (2016)

16. Jin, S., Lin, W., Yin, H., Yang, S., Li, A., Deng, B.: Community structure mining in big data social media networks with MapReduce. Cluster Comput. **18**(3), 999–1010 (2015)
17. Liu, H., Chen, L., Zhu, H., Lu, T., Liang, F.: Uncertainty community detection in social networks. J. Softw. **9**(4), 1045–1049 (2014)
18. Kang, Y., Yu, B., Wang, W., Meng, D.: Spectral clustering for large-scale social networks via a pre-coarsening sampling based NystrÖm method. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015, Part II. LNCS (LNAI), vol. 9078, pp. 106–118. Springer, Cham (2015). doi:10.1007/978-3-319-18032-8_9
19. Leung, C.K.-S., Cuzzocrea, A., Jiang, F.: Discovering frequent patterns from uncertain data streams with time-fading and landmark models. In: Hameurlain, A., Küng, J., Wagner, R., Cuzzocrea, A., Dayal, U. (eds.) TLDKS VIII. LNCS, vol. 7790, pp. 174–196. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37574-3_8
20. Leung, C.K.-S., Jiang, F.: Big data analytics of social networks for the discovery of "following" patterns. In: Madria, S., Hara, T. (eds.) DaWaK 2015. LNCS, vol. 9263, pp. 123–135. Springer, Cham (2015). doi:10.1007/978-3-319-22729-0_10
21. Leung, C.K.-S., Jiang, F., Pazdor, A.G.M., Peddle, A.M.: Parallel social network mining for interesting 'following' patterns. Concurr. Comput. Practice Exp. **28**(15), 3994–4012 (2016)
22. Leung, C.K.-S., MacKinnon, R.K.: BLIMP: a compact tree structure for uncertain frequent pattern mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 115–123. Springer, Cham (2014). doi:10.1007/978-3-319-10160-6_11
23. Leung, C.K.-S., MacKinnon, R.K., Tanbeer, S.K.: Fast algorithms for frequent itemset mining from uncertain data. In: IEEE ICDM 2014, pp. 893–898 (2014)
24. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 653–661. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68125-0_61
25. Leung, C.K.-S., Tanbeer, S.K.: Mining popular patterns from transactional databases. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 291–302. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32584-7_24
26. Leung, C.K.-S., Tanbeer, S.K., Cameron, J.J.: Interactive discovery of influential friends from social networks. Soc. Netw. Anal. Min. **4**(1), art. 154 (2014)
27. Ma, L., Huang, H., He, Q., Chiew, K., Wu, J., Che, Y.: GMAC: a seed-insensitive approach to local community detection. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 297–308. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40131-2_26
28. Madden, S.: From databases to big data. IEEE Internet Comput. **16**(3), 4–6 (2012)
29. Mumu, T.S., Ezeife, C.I.: Discovering community preference influence network by social network opinion posts mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 136–145. Springer, Cham (2014). doi:10.1007/978-3-319-10160-6_13
30. Rader, E., Gray, R.: Understanding user beliefs about algorithmic curation in the Facebook news feed. In: ACM CHI 2015, pp. 173–182 (2015)
31. Rajadesingan, A., Zafarani, R., Liu, H.: Sarcasm detection on Twitter: a behavioral modeling approach. In: ACM WSDM 2015, pp. 97–106 (2015)
32. Tanbeer, S.K., Leung, C.K.-S., Cameron, J.J.: Interactive mining of strong friends from social networks and its applications in e-commerce. J. Organ. Comput. Electron. Commerce **24**(2–3), 157–173 (2014)

33. Wang, Y., Vasilakos, A.V., Ma, J., Xiong, N.: On studying the impact of uncertainty on behavior diffusion in social networks. IEEE Trans. Syst. Man Cybern. Syst. **45**(2), 185–197 (2015)
34. Wei, E.H.-C., Koh, Y.S., Dobbie, G.: Finding maximal overlapping communities. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 309–316. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40131-2_27
35. Yu, W., Coenen, F., Zito, M., Salhi, S.: Minimal vertex unique labelled subgraph mining. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 317–326. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40131-2_28
36. Yuan, N.J.: Mining social and urban big data. In: ACM WWW 2015, p. 1103 (2015)