Finding Diverse Friends in Social Networks

Syed Khairuzzaman Tanbeer and Carson Kai-Sang Leung*

Department of Computer Science, University of Manitoba, Canada
kleung@cs.umanitoba.ca

Abstract. Social networks are usually made of users linked by friendship, which can be dependent on (or influenced by) user characteristics (e.g., connectivity, centrality, weight, importance, activity in the networks). Among many friends of these social network users, some friends are more diverse (e.g., more influential, prominent, and/or active in a wide range of domains) than other friends in the networks. Recognizing these diverse friends can provide valuable information for various real-life applications when analyzing and mining huge volumes of social network data. In this paper, we propose a tree-based mining algorithm that finds diverse friends, who are highly influential across multiple domains, in social networks.

1 Introduction and Related Works

Social networks [9,11] are made of social entities (e.g., users) who are linked by some specific types of relationships (e.g., friendship, common interest, kinship). Facebook, Google+, LinkedIn, Twitter and Weibo [13,16] are some examples of social networks. Within these networks, a user f_i usually can create a personal profile, add other users as friends, endorse their skills/expertise, exchange messages among friends. These social networks may consist of thousands or millions of users, and each user f_i can have different number of friends. Among them, some are more important or influential than others [2,6,7,15,17].

Over the past few years, several data mining techniques [5,8,12,14] have been developed to help users extract implicit, previously unknown, and potentially useful information about the important friends. Recent works on social network mining include the discovery of strong friends [3] and significant friends [10] based on the degree of one-to-one interactions (e.g., based on the number of postings to a friend's wall).

However, in some situations, it is also important to discover users who (i) are influential in the social networks, (ii) have high level of expertise in some domains, and/or (iii) have diverse interest in multiple domains. In other words, users may want to find important friends based on their influence, prominence, and/or diversity. For instance, some users may be narrowly interested in one specific domain (e.g., computers). Other users may be interested in a wide range of domains (e.g., computers, music, sports), but their expertise level may vary from one domain to another (e.g., a user f_i may be a computer expert but only a beginner in music).

* Corresponding author.

Y. Ishikawa et al. (Eds.): APWeb 2013, LNCS 7808, pp. 301–309, 2013.

Table 1. Prominence values & lists of interest groups

(a) Prominence of friends

Friend (f_i)	Prominence $Prom(f_i)$		
	Domain D_1	Domain D_2	Domain D_3
Amy	0.45	0.60	0.50
Bob	0.90	0.70	0.30
$Cathy$	0.20	0.60	0.70
Don	0.30	0.50	0.40
Ed	0.50	0.40	0.45
$Fred$	0.42	0.24	0.70
$Greg$	0.57	0.10	0.20

(b) Lists of interest groups in F_{SN}

Domain	Interest-group list L_j
D_1	$L_1 = \{Amy, Bob, Don\}$ $L_2 = \{Cathy, Don\}$ $L_3 = \{Amy, Bob\}$
D_2	$L_4 = \{Bob, Greg\}$ $L_5 = \{Bob, Cathy, Don\}$ $L_6 = \{Cathy, Ed\}$ $L_7 = \{Bob, Cathy, Ed\}$
D_3	$L_8 = \{Amy, Cathy, Ed\}$ $L_9 = \{Amy, Fred\}$ $L_{10} = \{Amy, Cathy\}$

In this paper, one of our *key contributions* is an efficient tree-based algorithm called **Div-growth** for mining diverse friends from social networks. Div-growth takes into account multiple properties (e.g., influence, prominence, and/or diversity) of friends in the networks. Another *key contribution* is a prefix-tree based structure called **Div-tree** for capturing the social network data in a memory-efficient manner. Once the Div-tree is constructed, Div-growth computes the diversity of users based on both their influence and prominence to mine diverse groups of friends.

The remainder of this paper is organized as follows. We introduce the notion of diverse friends in the next section. Section 3 introduces our Div-growth algorithm, which mines diverse friends from our Div-tree. Experimental results are reported in Section 4; conclusions are presented in Section 5.

2 Notion of Groups of Diverse Friends

Consider a social network on three different domains (domains D_1, D_2, D_3) and seven individuals (*Amy, Bob, Cathy, Don, Ed, Fred* & *Greg*) with prominence values in each domain, as shown in Table 1(a). Each *domain* represents a sub-category (e.g., sports, arts, education) of interest. The **prominence value** of an individual reveals his level of expertise (e.g., importance, weight, value, reputation, belief, position, status, or significance) in a domain. In other words, the prominence value indicates how important, valued, significant, or well-positioned the individual is in each domain. The prominence value can be measured by using a common scale, which could be (i) specified by users or (ii) automatically calculated based on some user-centric parameters (e.g., connectivity, centrality, expertise in the domain, years of membership in the domain, degree of involvement in activities in the domain, numbers of involved activities in the domain). In this paper, the prominence value is normalized into the range (0, 1]. As the same individual may have different levels of expertise in different domains, his corresponding prominence value may vary from one domain to another. For example, prominence value $Prom_{D_1}(Amy)$ of Amy in domain D_1 is 0.45, which (i) is different from $Prom_{D_2}(Amy)$=0.60 and (ii) is higher than $Prom_{D_1}(Cathy)$=0.20 (implying that Amy is more influential than $Cathy$ in D_1).

Consistent with existing settings of a social network [3,5,10], let $F = \{f_1, f_2, \ldots, f_m\}$ be a set of individuals/friends in a social network. An *interest-group list* $L \subseteq F$ is a list of individuals who are connected as friends due to some common interests. Let $G = \{f_1, f_2, \ldots, f_k\} \subseteq F$ be a **group of friends** (i.e., friend group) with k friends. Then, $Size(G) = k$, which represents the number of individuals in G. A *friend network* $F_{SN} = \{L_1, L_2, \ldots, L_n\}$ is the set of all n interest-group lists in the entire social network. These lists belong to some domains, and each domain contains at least one list. The set of lists in a particular domain D is called a *domain database* (denoted as F_D). Here, we assume that there exists an interest-group list in every domain. The *projected list* F_D^G of G in F_D is the set of lists in F_D that contains group G. The frequency $Freq_D(G)$ of G in F_D indicates the number of lists L_j's in F_D^G, and the frequencies of G in multiple domains are represented as $Freq_{D_{1,2,\ldots,d}}(G) = \langle Freq_{D_1}(G), Freq_{D_2}(G), \ldots, Freq_{D_d}(G)\rangle$.

Example 1. Consider F_{SN} shown in Table 1(b), which consists of n=10 interest-group lists $L_1, ..., L_{10}$ for m=7 friends in Table 1(a). Each row in the table represents the list of an interest group. These 10 interest groups are distributed into d=3 domains D_1, D_2 and D_3. For instance, $F_{D_1} = \{L_1, L_2, L_3\}$. For group $G = \{Cathy, Ed\}$, its $Size(G)$=2. As its projected lists on the 3 domains are $F_{D_1}^G$=$\emptyset$, $F_{D_2}^G$=$\{L_6, L_7\}$ and $F_{D_3}^G$=$\{L_8\}$, its frequencies $Freq_{D_{1,2,3}}(G) = \langle 0, 2, 1\rangle$. □

Definition 1. The **prominence value** $\boldsymbol{Prom_D(G)}$ of a friend group G in a single domain D is defined as the average of all prominence values for all the friends in G: $\boldsymbol{Prom_D(G)} = \frac{\sum_{i=1}^{Size(G)} Prom_D(f_i)}{Size(G)}$. Prominence values of G in multiple domains are represented as $Prom_{D_{1,2,\ldots,d}}(G) = \langle Prom_{D_1}(G), Prom_{D_2}(G), \ldots, Prom_{D_d}(G)\rangle$. □

Definition 2. The **influence** $\boldsymbol{Inf_D(G)}$ of a friend group G in a domain D in F_D is defined as the product of the prominence value of G in the domain D and its frequency in the domain database F_D, i.e., $\boldsymbol{Inf_D(G) = Prom_D(G) \times Freq_D(G)}$. For multiple domains, $Inf_{D_{1,2,\ldots,d}}(G) = \langle Inf_{D_1}(G), Inf_{D_2}(G), \ldots, Inf_{D_d}(G)\rangle$. □

Definition 3. The **diversity** $\boldsymbol{Div(G)}$ of a friend group G among all d domains in F_{SN} is defined as the average of all the influence values of G in all domains in the social network: $\boldsymbol{Div(G)} = \frac{\sum_{j=1}^{d} Inf_{D_j}(G)}{d}$. □

Example 2. Revisit F_{SN} in Table 1(b). The prominence value of friend group $G = \{Cathy, Ed\}$ in $D_1 = \frac{Prom_{D_1}(Cathy)+Prom_{D_1}(Ed)}{Size(G)} = \frac{0.20+0.50}{2} = 0.35$. We apply similar computation and get $Prom_{D_{1,2,3}}(G) = \langle 0.35, \frac{0.60+0.40}{2}, \frac{0.70+0.45}{2}\rangle = \langle 0.35, 0.5, 0.575\rangle$. Recall from Example 1 that $Freq_{D_{1,2,3}}(G) = \langle 0, 2, 1\rangle$. So, the overall influence of G in all 3 domains can be calculated as $Inf_{D_{1,2,3}}(G) = \langle 0.35\times 0, 0.5\times 2, 0.575\times 1\rangle = \langle 0, 1, 0.575\rangle$. Thus, the diversity of G in these d=3 domains in F_{SN} is $Div(G)$=$\frac{0+1+0.575}{3}$=0.525. □

A group G of friends in a social network F_{SN} is considered **diverse** if its diversity value $Div(G) \geq$ the user-specified minimum threshold *minDiv*, which can be expressed as an absolute (non-negative real) number or a relative percentage (with respect to the size of F_{SN}). Given F_{SN} and *minDiv*, the research problem of ***mining diverse friends from social networks*** is to find every group G of friends having $Div(G) \geq minDiv$.

Example 3. Recall from Example 2 that $Div(\{Cathy, Ed\})$=0.525. Given (i) F_{SN} in Table 1(b) and (ii) the user-specified $minDiv$=0.5, group G=$\{Cathy, Ed\}$ is *diverse* because $Div(G)$=0.525 $\geq$ 0.5=$minDiv$. But, group G'=$\{Ed\}$ is *not* diverse because $Div(G') = \frac{(0.5\times0)+(0.4\times2)+(0.45\times1)}{3} = \frac{0+0.8+0.45}{3} = 0.417 < minDiv$. □

3 Our Div-growth Algorithm for Mining Diverse Friends

When mining frequent patterns, the frequency/support measure [1,4] satisfies the downward closure property (i.e., all supersets of an infrequent patterns are infrequent). This helps reduce the search/solution space by pruning infrequent patterns, which in turn speeds up the mining process. However, when mining diverse friends, diversity does *not* satisfy the downward closure property. Recall from Example 3, group G'=$\{Ed\}$) is not diverse but its super-group G=$\{Cathy, Ed\}$) is diverse. As we cannot prune those groups that are not diverse, the mining of diverse friends can be challenging.

To handle this challenge, for each domain D, we identify the **(global) maximum prominence value $GMProm_D$** among all friends. Then, for each friend f_i, we calculate an upper bound of the influence value $Inf^U_D(f_i)$ by multiplying $GMProm_D$ (instead of the actual $Prom_D(f_i)$) with the corresponding frequency $Freq_D(f_i)$. The upper bound of diversity value $Div^U(f_i)$ can then be computed by using $Inf^U_D(f_i)$.

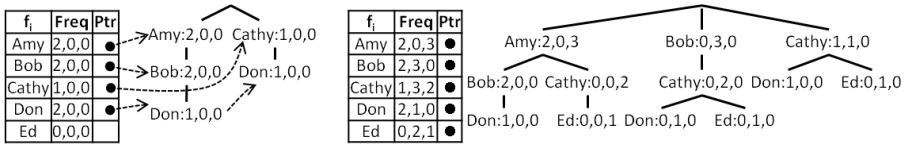
Lemma 1. Let G be a group of friends in F_{SN} such that a friend $f_i \in G$. If $Div^U(f_i) < minDiv$, then $Div(G)$ must also be less than $minDiv$. □

Example 4. Revisit F_{SN} in Table 1(b). Note that $GMProm_{D_1}$=0.90, $GMProm_{D_2}$= 0.70, and $GMProm_{D_3}$=0.70. Recall from Example 3 that $Freq_{D_{1,2,3}}(\{Ed\})=\langle 0, 2, 1\rangle$. Then, we can compute $Div^U(\{Ed\})=\frac{(0.90\times0)+(0.70\times2)+(0.70\times1)}{3}$=0.7 $\geq$ $minDiv$. So, we do not prune $\{Ed\}$ to avoid missing its super-group $\{Cathy, Ed\}$, which is diverse. Similarly, we can compute $Div^U(\{Fred\})=\frac{(0.90\times0)+(0.70\times0)+(0.70\times1)}{3}$=0.23 < $minDiv$. Due to Lemma 1, we prune *Fred* as none of its super-groups can be diverse. □

3.1 Phase 1: Constructing a Div-tree Structure

Given F_{SN} and $minDiv$, our proposed Div-growth algorithm constructs a Div-tree as follows. It first scans F_{SN} to calculate $Freq_{D_j}(f_i)$ for each friend f_i in each domain D_j. For each f_i, Div-growth then uses $GMProm_D$ to compute the upper bound of the diversity value $Div^U(f_i)$, which is used to prune groups of friends who are not potentially diverse. Every potentially diverse friend f_i, along with its $Freq_{D_{1,\ldots,d}}(f_i)$, is stored in the header table.

Afterwards, Div-growth scans F_{SN} the second time to capture the important information about potentially diverse friends in a user-defined order in the Div-tree. Each tree node consists of (i) a friend name and (ii) its frequency counters for all d domains in the respective path. The basic construction process of a Div-tree is similar to that of the FP-tree [4]. A key difference is that, rather than using only a single frequency counter capturing either the maximum or average frequency for all domains (which may lead to loss of information), we use d frequency counters capturing the frequency for all d domains. See Example 5.



(a) After inserting L_1 to L_3 in D_1 **(b)** After inserting all lists in all 3 domains

Fig. 1. Construction of a Div-tree

Example 5. To construct a Div-tree for F_{SN} shown in Table 1(b) when $minDiv$=0.5, Div-growth scans F_{SN} to compute (i) $GMProm_{D_{1,2,3}} = \langle 0.9, 0.7, 0.7\rangle$ for all d=3 domains, (ii) frequencies of all 7 friends in d=3 domains (e.g., $Freq_{D_{1,2,3}}(\{Amy\}) = \langle 2, 0, 3\rangle$), (iii) upper bound of diversity values of all 7 friends (e.g., $Div^U(\{Amy\}) = \frac{(0.9\times 2)+(0.7\times 0)+(0.7\times 3)}{3} = 1.3$ using $Inf^U_{D_{1,2,3}}(\{Amy\})$). Based on Lemma 1, we safely remove $Fred$ and $Greg$ having $Div^U(\{Fred\})$=0.23 and $Div^U(\{Greg\})$=0.23 both below $minDiv$ as their super-groups cannot be diverse. So, the header table includes only the remaining 5 friends—sorted in some order (e.g., lexicographical order of friend names)—with their $Freq_{D_{1,2,3}}(\{f_i\})$. To facilitate a fast tree traversal, like the FP-tree, the Div-tree also maintains horizontal node traversal pointers from the header table to nodes of the same f_i.

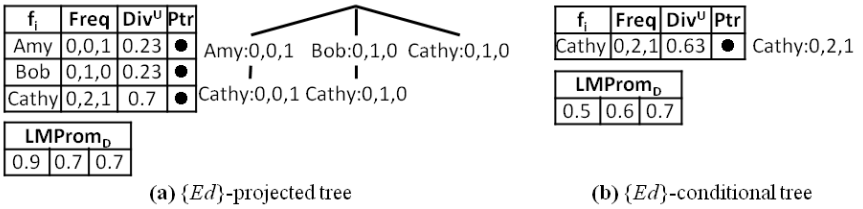
Div-growth then scans each $L_j \in F_{SN}$, removes any friend $f_i \in L_j$ having $Div^U(f_i) < minDiv$, sorts the remaining friends according to the order in the header table, and inserts the sorted list into the Div-tree. Each tree node captures (i) f_i representing the group G consisting of all friends from the root to f_i and (ii) its frequencies in each domain $Freq_{D_{1,2,3}}(G)$. For example, the rightmost node Ed:0,1,0 of the Div-tree in Fig. 1(b) captures G=$\{Cathy, Ed\}$ and $Freq_{D_{1,2,3}}(G)$=$\langle 0, 1, 0\rangle$. Tree paths of common prefix (i.e., same friends) are shared, and their corresponding frequencies are added. See Figs. 1(a) and 1(b) for Div-trees after reading all interest-group lists in domain D_1 and the entire F_{SN}, respectively. □

With this tree construction process, the size of the Div-tree for F_{SN} with a given $minDiv$ is observed to be bounded above by $\sum_{L_j \in F_{SN}} |L_j|$.

3.2 Phase 2: Mining Diverse Friend Groups

Once the Div-tree is constructed, Div-growth recursively mines diverse friend groups by building projected and conditional trees in a fashion similar to that of FP-growth [4].

Recall that $Div(G)$ computed based on $Prom_D(G)$ does *not* satisfy the downward closure property. To facilitate pruning, we use $GMProm_D(f_i)$ to compute $Div^U(f_i)$, which then satisfies the downward closure property. However, if $Div^U(G)$ was computed as an upper bound to super-group G of f_i, then it may overestimate diversity of G and may lead to false positives. To reduce the number of false positives, Div-growth uses the **local maximum prominence value** $\boldsymbol{LMProm_D(G)} = \max_{f_i \in F^G_D}\{Prom_D(G)\}$ for the projected and conditional trees for G. See Lemma 2 and Example 6.



(a) $\{Ed\}$-projected tree **(b)** $\{Ed\}$-conditional tree

Fig. 2. Tree-based mining of diverse friend groups

Lemma 2. The diversity value of a friend group G computed based on $LMProm_D(G)$ is a tighter upper bound than $Div^U(G)$ computed based on $GMProm_D$. □

Example 6. To mine potentially diverse friend groups from the Div-tree in Fig. 1(b) using $minDiv$ =0.5, Div-growth first builds the $\{Ed\}$-projected tree—as shown in Fig. 2(a)—by extracting the paths ⟨Amy, Cathy, Ed⟩:0,0,1, ⟨Bob, Cathy, Ed⟩:0,1,0 and ⟨Cathy, Ed⟩:0,1,0 from the Div-tree in Fig. 1(b). For $F^{Ed}_{D_{1,2,3}}$=$\{Amy, Bob, Cathy, Ed\}$, Div-growth also uses $LMProm_{D_{1,2,3}}(F^{Ed}_{D_{1,2,3}}) = \langle 0.9, 0.7, 0.7\rangle$ to compute the tightened $Div^U(G)$ such as tightened $Div^U(\{Amy, Ed\})$=$\frac{(0.9\times0)+(0.7\times0)+(0.7\times1)}{3}$=0.23 < $minsup$.

As $Div^U(\{Amy, Ed\})$ and $Div^U(\{Bob, Ed\})$ are both below $minsup$, Div-growth prunes Amy and Bob from the $\{Ed\}$-projected tree to get the $\{Ed\}$-conditional tree as shown in Fig. 2(b). Due to pruning, Div-growth recomputes $LMProm_{D_{1,2,3}}(F^{Ed}_{D_{1,2,3}})$ =$\langle 0.5, 0.6, 0.7\rangle$ and the tightened $Div^U(\{Cathy, Ed\})$=$\frac{(0.5\times0)+(0.6\times2)+(0.7\times1)}{3}$=0.63 for the updated $F^{Ed}_{D_{1,2,3}}$=$\{Cathy, Ed\}$. This completes the mining for $\{Ed\}$.

Next, Div-growth builds $\{Don\}$-, $\{Cathy\}$- & $\{Bob\}$-projected and conditional trees, from which potentially diverse friend groups can be mined. Finally, Div-growth computes the true diversity value $Div(G)$ for each of these mined groups to check if it is truly diverse (i.e., to remove all false positives). □

4 Experimental Results

To evaluate the effectiveness of our proposed Div-growth algorithm and its associated Div-tree structure, we compared them with a closely related *weighted* frequent pattern mining algorithm called Weight [18] (but it does not use different weights for individual items). As Weight was designed for frequent pattern mining (instead of social network mining), we apply those datasets commonly used in frequent pattern mining for a fair comparison: (i) IBM synthetic datasets (e.g., T10I4D100K) and (ii) real datasets (e.g., mushroom, kosarak) from the Frequent Itemset Mining Dataset Repository `fimi.cs.helsinki.fi/data`. See Table 2 for more detail. Items in transactions in these datasets are mapped into friends in interest-group lists. To reflect the concept of *domains*, we subdivided the datasets into several batches. Moreover, a random number in the range (0, 1] is generated as a prominence value for each friend in every domain.

All programs were written in C++ and run on the Windows XP operating system with a 2.13 GHz CPU and 1 GB main memory. The runtime specified indicates the total execution time (i.e., CPU and I/Os). The reported results

Table 2. Dataset characteristics

Dataset	#transactions	#items	maxL	avgTL	Density
mushroom	8,124	119	23	23.0	Dense
T10I4D100K	100,000	870	29	10.1	Sparse
kosarak	990,002	41,270	2498	8.1	Sparse

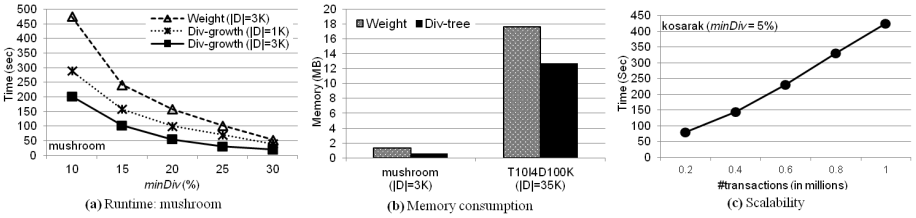


Fig. 3. Experimental results

are based on the average of multiple runs for each case. We obtained consistent results for all of these datasets.

Runtime. First, we compared the runtime of Div-growth (which includes the construction of the Div-tree, the mining of potentially diverse friend groups from the Div-tree, and the removal of false positives) with that of Weight. Fig. 3(a) shows the results for a dense dataset (mushroom), which were consistent with those for sparse datasets (e.g., T10I4D100K). Due to page limitation, we omit the results for sparse datasets. Runtimes of both algorithms increased when mining larger datasets (social networks), more batches (domains), and/or with lower *minDiv* thresholds. Between the two algorithms, our tree-based Div-growth algorithm outperformed the Apriori-based Weight algorithm. Note that, although FP-growth [4] is also a tree-based algorithm, it was *not* design to capture weights. To avoid distraction, we omit experimental results on FP-growth and only show those on Weight (which captures weights).

Compactness of the Div-tree. Next, we evaluated the memory consumption. Fig. 3(b) shows the amount of memory required by our Div-tree for capturing the content of social networks with the lowest *minDiv* threshold (i.e., without removing any friends who were not diverse). Although this simulated the worst-case scenario for our Div-tree, Div-tree was observed (i) to consume a reasonable amount of memory and (ii) to require less memory than Weight (because our Div-tree is compact due to the prefix sharing).

Scalability. Then, we tested the scalability of our Div-growth algorithm by varying the number of transactions (interest-group lists). We used the kosarak dataset as it is a huge sparse dataset with a large number of distinct items (individual users). We divided this dataset into five portions, and each portion is subdivided into multiple batches (domains). We set *minDiv*=5% of each portion. Fig. 3(c) shows that, when the size of the dataset increased, the runtime also increased proportionally implying that Div-growth is scalable.

Additional Evaluation. So far, we have evaluated the *efficiency* (e.g., runtime, compactness or memory consumption, as well as scalability) of our Div-growth algorithm. Experimental results show that Div-growth is time- and space-efficient as well as scalable. As ongoing work, we plan to evaluate the *quality* (e.g., precision) of Div-growth in finding diverse friend groups. Moreover, for a fair comparison with Weight, we have used those datasets that are commonly used in frequent pattern mining. As ongoing work, we plan to evaluate Div-growth using real-life social network datasets.

5 Conclusions

In this paper, we (i) introduced a new notion of *diverse friends* for social networks, (ii) proposed a compact tree structure called *Div-tree* to capture important information from social networks, and (iii) designed a tree-based mining algorithm called *Div-growth* to find diverse (groups of) friends from social networks. Diversity of friends is measured based on their prominence, frequency and influence in different domains on the networks. Although diversity does not satisfy the downward closure property, we managed to address this issue by using the global and local maximum prominence values of users as upper bounds. Experimental results showed that (i) our Div-tree is compact and space-effective and (ii) our Div-growth algorithm is fast and scalable for both sparse and dense datasets. As ongoing work, we conduct more extensive experimental evaluation to measure other aspects (e.g., precision) of our Div-growth algorithm in finding diverse friends. We also plan to (i) design a more sophisticated way to measure influence and (ii) incorporate other computational metrics (e.g., popularity, significance, strength) with prominence into our discovery of useful information from social networks.

Acknowledgements. This project is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994, pp. 487–499 (1994)
2. Anagnostopoulos, A., Kumar, R., Mahdian, M.: Influence and correlation in social networks. In: ACM KDD 2008, pp. 7–15 (2008)
3. Cameron, J.J., Leung, C.K.-S., Tanbeer, S.K.: Finding strong groups of friends among friends in social networks. In: IEEE DASC/SCA 2011, pp. 824–831 (2011)
4. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12 (2000)
5. Jiang, F., Leung, C.K.-S., Tanbeer, S.K.: Finding popular friends in social networks. In: CGC/SCA 2012, pp. 501–508. IEEE (2012)
6. Kamath, K.Y., Caverlee, J., Cheng, Z., Sui, D.Z.: Spatial influence vs. community influence: modeling the global spread of social media. In: ACM CIKM 2012, pp. 962–971 (2012)

7. Lee, W., Leung, C.K.-S., Song, J.J., Eom, C.S.-H.: A network-flow based influence propagation model for social networks. In: CGC/SCA 2012, pp. 601–608. IEEE (2012)
8. Lee, W., Song, J.J., Leung, C.K.-S.: Categorical data skyline using classification tree. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, M.A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 181–187. Springer, Heidelberg (2011)
9. Leung, C.K.-S., Carmichael, C.L.: Exploring social networks: a frequent pattern visualization approach. In: IEEE SocialCom 2010, pp. 419–424 (2010)
10. Leung, C.K.-S., Tanbeer, S.K.: Mining social networks for significant friend groups. In: Yu, H., Yu, G., Hsu, W., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA Workshops 2012. LNCS, vol. 7240, pp. 180–192. Springer, Heidelberg (2012)
11. Peng, Z., Wang, C., Han, L., Hao, J., Ou, X.: Discovering the most potential stars in social networks with infra-skyline queries. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 134–145. Springer, Heidelberg (2012)
12. Sachan, M., Contractor, D., Faruquie, T.A., Subramaniam, L.V.: Using content and interactions for discovering communities in social networks. In: ACM WWW 2012, pp. 331–340 (2012)
13. Schaal, M., O'Donovan, J., Smyth, B.: An analysis of topical proximity in the twitter social graph. In: Aberer, K., Flache, A., Jager, W., Liu, L., Tang, J., Guéret, C. (eds.) SocInfo 2012. LNCS, vol. 7710, pp. 232–245. Springer, Heidelberg (2012)
14. Sun, Y., Barber, R., Gupta, M., Aggarwal, C.C., Han, J.: Co-author relationship prediction in heterogeneous bibliographic networks. In: ASONAM 2011, pp. 121–128. IEEE (2011)
15. Tanbeer, S.K., Leung, C.K.-S., Cameron, J.J.: DIFSoN: discovering influential friends from social networks. In: CASoN 2012, pp. 120–125. IEEE (2012)
16. Yang, X., Ghoting, A., Ruan, Y., Parthasarathy, S.: A framework for summarizing and analyzing twitter feeds. In: ACM KDD 2012, pp. 370–378 (2012)
17. Zhang, C., Shou, L., Chen, K., Chen, G., Bei, Y.: Evaluating geo-social influence in location-based social networks. In: ACM CIKM 2012, pp. 1442–1451 (2012)
18. Zhang, S., Zhang, C., Yan, X.: Post-mining: maintenance of association rules by weighting. Information Systems 28(7), 691–707 (2003)