Recommending Items in Social Networks using Cliques-Based Trust

Lidia Fotia†
† DIIES, University Mediterranea of Reggio Calabria, Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy, lidia.fotia@unirc.it

***Abstract*—Thematic groups are gaining a lot of attention and high centrality in online community, as users share opinions and/or mutually collaborate for reaching their targets. In this paper we consider the concept of clique in the social networks. A clique is a group of actors connected to each other more closely than the overall network of which they are a part. In the common language, the term clique means an informal and restricted social group formed by people who share common interests or characteristics. This paper proposes a new trust measure in social networks which focuses on cliques. In particular, we represent this scenario by means of a specific model, and we introduce an algorithm for detecting trust recommendations. We show a complete example of how our approach works.**

***Index Terms*—Recommendation, Online Communities, Trust, Group.**

I. Introduction

In the last few years, the number of users of online social networks has grown at an unprecedented rate. Many social networks provide tools which recommend friends, and sometimes resources, to their users. These tools rely on a notion of similarity between users or between a user and a resource. Obviously, they are not able to recommend to a user new social networks potentially relevant to him.

For this reason, an important target in Online Social Networks (OSNs) is to implement recommender systems capable to supply users with profitable suggestions about users to contact as interlocutors or interesting content to access. To achieve this goal it is necessary to consider the opinions provided by the community about the other users or the OSN content [1]–[3]. However, some users have malicious or even fraudulent behaviours in OSNs, thus the recommender systems could generate unreliable recommendations. Therefore, in these contexts, it is necessary to introduce the concept of trust.

The issue of trusting interlocutors largely emerged in online e-Commerce communities as, for instance eBay, and now it is discussed in many OSNs which allow their users to create and share contents with other users as well as opinions. For example, in the social networks EPINIONS[1] and CIAO[2] the users supply reviews about commercial products of different categories. Almost all of these platforms tackle this goal by adopting a reputation system.

Reputation is a form of indirect trust, where a user takes advantage from the opinions coming from other users for evaluating the level of trustworthiness of an interlocutor. Commonly, in the traditional OSN contexts, the reputation of a user is evaluated by averaging feedbacks provided by all the other users belonging to the same community.

In the past literature, there are many approaches based on global reputation [4]–[6] which is calculated on the evaluation of the users' behaviours shared with the entire community, and local reputation [7] that is based on the recommendations coming by the entourage of the user (friends, friends of friends and so on). The first type of approaches shows a limitation due to the difficulty of taking into account the effects of malicious or fraudulent behaviours, thus making the feedback themselves. Also, they are limited by the fact they need a training phase in generating recommendations. The latter overcome these limitations but they do not consider a group-based structure [8]–[10].

In this paper we present an approach to find a certain class of groups of users linked by trust relationships. In particular, we apply the BronKerbosch algorithm [11] to find the cliques in a social network. The algorithm is a simple backtracking procedure that recursively solves subproblems specified by three sets of vertices: the vertices that are required to be included in a partial clique, the vertices that are to be excluded from the clique, and some remaining vertices whose status still needs to be determined. Moreover, our approach depends on two main parameters: the relevance given to the reliability with respect to the reputation and the threshold of recommendation under which a product can be considered as not interesting. We propose an algorithm for detecting trust recommendations for a user considering the recommendations that come from users within his own cliques and those of other cliques weighted with the global reputation. We also present a complete example of how our approach works and an application to a real social network.

The paper is organised as follows. In the next section, we describe the BronKerbosch algorithm. In Section III, we provides technical details about our approach for finding trust recommendations about products. In Section IV, we survey the related work of the recent literature. In Sections V and VI, we describe a possible case study of the application. Finally, in Section VII we draw our conclusions and illustrate some possible future works.

II. The BronKerbosch Algorithm

The BronKerbosch algorithm [11] is a recursive backtracking algorithm used to find all maximal cliques in a graph. A

[1]www.epinions.com
[2]www.ciao.it

recursive call to the BronKerbosch algorithm supplies three sets of vertices C, S, and B as arguments, where C is a (possibly non-maximal) clique and $S \cup B = \Gamma(C)$ are the vertices that are adjacent to every vertex in C. Within the recursive call, the algorithm lists all cliques in $S \cup C$ that are maximal within the subgraph induced by $S \cup C \cup B$.

The algorithm selects $v \in S$ to add to the clique C, and makes a recursive call in which v has been moved from C to S. In this phase, it restricts B to the neighbors of v because non-neighbours cannot affect the maximality of the resulting cliques. At the end of the recursive call, v is moved to B. The recursion procedure ends when S and B are empty and the output C is a maximal clique. To obtain all maximal cliques in the graph, This recursive algorithm is called with S equal to the set of all vertices in the graph and with C and B empty. The algorithm is shown in Listing 1.

Listing 1. BronKerbosch algorithm
```
proc BronKerbosch(C, S, B)
1: if ($S \cup B$ = 0) then
2:     report C as a maximal clique
3: end if
4: for each vertex $v \in S$ do
5:     BronKerbosch($S \cap \Gamma(v)$, $C \cup \{v\}$, $B \cap \Gamma(v)$)
6:     $S := S \backslash \{v\}$
7:     $B := B \cup \{v\}$
8: end for
```

III. Our scenario

We introduce a virtual community V, formally denoted as $V = \langle \mathcal{A}, \mathcal{C} \rangle$, where $\mathcal{A}$ is the set of *agents* joined with V and $\mathcal{C}$ is the set of *cliques* contained in V. Moreover, each clique c is handled by an administrator agent a_c. All such communities are organised in social structures based on social relationship (like, Facebook or Twitter). The formation of a group (i.e. cliques) is a process based on two main events: a user asks for joining with a group and the administrator of the group accepts or refuses the request.

In our proposal, we introduce two measures to ensure that the recommendation system generates correct outputs. The first measure is the trust. We denote it by the mapping $TRUST_{x,y}$ that receives as input two agents x and y and produces as output a value representing the degree of trust between two agents: $TRUST_{x,y}$ = 0 (resp. $TRUST_{x,y}$ = 1) means that x assigns the minimum (resp. maximum) trustworthiness to y. The trust measure is asymmetric. If x does not have a sufficient direct knowledge of y, the agent must use the recommendations coming from the other agents of the community. For this reason, we introduce a more general trust measure, by combining two components $REL_{x,y}$ and REP_y, where *(i)* $REL_{x,y}$ is the direct reliability of y because has interacted in the past with x while *(ii)* REP_y is the global reputation of y, i.e. the trustworthiness that the whole community has in y. In particular, $REL_{x,y}$ assumes values ranging in the domain $[0 \dots 1] \bigcup \{NULL\}$, while $REL_{x,y}$ = NULL means that y did not have past interactions with x and thus it is not able to evaluate y's trustworthiness. In order to compute the reputation, we adopt the notion of

$$REP_y = \frac{1}{h_{max}|REV_y|} \sum_{\rho \in REV_y} h_\rho \qquad (1)$$

where $|REV_y|$ is the set of the reviews made by the user y and h is the helpfulness, i.e., it is associated with each review and represents the level of satisfaction of the other users for that review. To normalise REP_y, we divide it by the maximum value of the helpfulness h_{max}. REP_y assumes values ranging in the interval $[0 \dots 1] \epsilon \mathbb{R}$. Finally, we define $TRUST_{x,y}$ of x about y in the interval $[0 \dots 1]$ as follows:

$$TRUST_{x,y} = \alpha \cdot REL_{x,y} + (1 - \alpha) \cdot REP_y \qquad (2)$$

where α is a real number, ranging in [0...1], which is set by x to weight the relevance he/she assigns to the reliability with respect to the reputation.

A. The Trust-Based Recommendation System without and with cliques

At this step, x receives some recommendations about the community products. In particular, REC_x^p is the recommendation that the user x receives about the product p. It is calculated as follows:

$$REC_x^p = \frac{\sum_{y \in A, y \neq x} TRUST_{x,y} \cdot rate_y^p}{\sum_{y \in A, y \neq x} TRUST_{x,y}} \qquad (3)$$

where $rate_y^p$ is the review of the the user y about the product p (a number between 1 and 6), weighed by the trust of y. This means that his/her opinion about a product is taken into account if his/her trustworthiness is high.

1) Cliques: Now, we introduce the clique's concept in the community. The cliques are calculated following the BronKerbosch algorithm described in the section II. In this case, we suppose that the trust perceived by an agent x with respect to the component of its clique is equal to $t_{x,y} + \Delta$, instead the agent x considers the trust that user has in the whole community (see Equation 2). In other words, if a user y belongs to the same clique of x, $t_{x,y}$ can be increased by a value Δ. Being part of the same clique, even if x and y do not know each other directly, they have interests in common and there is an indirect connection between them. Δ is a real number, ranging in [0...1], which is set by x to weight the relevance he/she assigns to the clique with respect to the standard configuration. $t_{x,y} + \Delta$ must not exceed 1.

IV. Related Work

Concerning the concept of trust, there are several proposals in the literature and in several domains [12]–[16] In particular, a large number of papers in the literature investigated on the topic we deal with here, therefore, in this section we cite only those approaches which we consider comparable with that discussed in this paper.

Sherchan et al. [17] present an important review of trust, in which they comprehensively examine trust definitions and

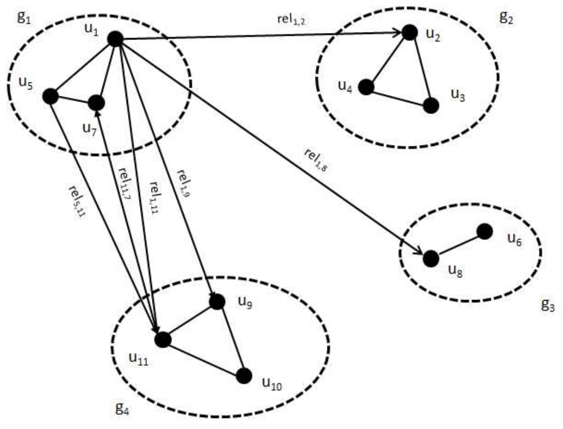

Fig. 1. The cliques obtained by applying the BronKerbosch algorithm.

measurements, from multiple fields including sociology, psychology, and computer science. Trust models [18]–[21] allow to exploit information derived by direct experiences and/or opinions of others to trust potential partners by means of a single measure [22], [23]. Xia et al. [24] build a subjective trust management model AFStrust, which considers multiple factors including direct trust, recommendation trust, incentive function and active degree, and treats those factors based on the analytic hierarchy process (AHP) theory and the fuzzy logic rule. [25] describes how to build robust reputation systems using machine learning techniques, and defines a framework for translating a trust modelling problem into a learning problem.

In many disciplines, there is a population of people which should be optimally divided into multiple groups based on certain attributes to collaboratively perform a particular task [26], [27]. The problem becomes more complex when some other requirements are also added: homogeneity, heterogeneity or a mixture of teams, amount of consideration to the preferences of individuals, variability or invariability of group size, having moderators, aggregation or distribution of persons, overlapping level of teams, and so forth [5], [28]–[30]. Basu et al. [31] consider the problem of how to form groups such that the users in the formed groups are most satisfied with the suggested top-k recommendations. They assume that the recommendations will be generated according to one of the two group recommendation semantics, called Least Misery and Aggregate Voting. Rather than assuming groups are given, or rely on ad hoc group formation dynamics, their framework allows a strategic approach for forming groups of users in order to maximise satisfaction. In [32], the authors reveal how these problems can be mathematically formulated through a binary integer programming approach to construct an effective model which is solvable by exact methods in an acceptable time.

V. An e-Commerce case study

Now, we explain how it is possible to use cliques to generate the recommendations of products for the users inserted into an online e-Commerce communities. As an example, we propose to model our community by a graph G.

We apply the BronKerbosch algorithm (see Section II) to the graph depicted in Figure 1, which also contains the final set of cliques obtained by the algorithm itself. In our case, there are four cliques called c_1, c_2, c_3 and c_4. Moreover, the values of REL are as follows: $REL_{1,2} = 0.6$, $REL_{1,5} = 1$, $REL_{1,7} = 0,82$, $REL_{1,8} = 0,4$, $REL_{1,9} = 0,3$, $REL_{1,11} = 0,1$, $REL_{2,3} = 0,81$, $REL_{2,4} = 0,81$, $REL_{3,2} = 0,9$, $REL_{3,4} = 0,85$, $REL_{4,2} = 0,9$, $REL_{4,3} = 0,81$, $REL_{5,1} = 0,9$, $REL_{5,7} = 0,85$, $REL_{5,11} = 0,4$, $REL_{6,8} = 0,9$, $REL_{7,1} = 0,81$, $REL_{7,5} = 0,95$, $REL_{8,6} = 0,81$, $REL_{9,10} = 0,86$, $REL_{9,11} = 0,85$, $REL_{10,9} = 0,91$, $REL_{10,11} = 0,85$, $REL_{11,9} = 0,86$, $REL_{11,7} = 0,3$ and $REL_{11,10} = 0,9$.

In particular, in our example, there are nine products that are divided in three main categories called Electronics, Informatics and Software, these are shown in Table I, while in Table II, we show an example of dataset.

productID	name	categoryID
1	Car TomTom, Display 5”	Electronics
2	Smartphone Android 5.1	Electronics
3	TV HD Ready 15,6” Format 16:9	Electronics
4	Notebook 15” i7, RAM 8 GB, HDD 500GB	Informatics
5	Black and white laser printer	Informatics
6	Tablet 7”, Wi-Fi, 8 GB	Informatics
7	Microsoft Windows 7 PRO SP1 32/64-bit	Software
8	Microsoft Office 365 Personal - 32/64 Bit	Software
9	Nuance Power PDF Standard	Software

TABLE I
List of products

In our model, we have associated with each agent a profile contained, as unique feature, the reputation to review the products. This reputation has been computed by averaging, on all the reviews posted by the agent, the helpfulness associated with each review, where the helpfulness is an information available on the dataset and obtained by the opinions expressed by the users of the community. We obtain that the reputation values (see Equation 1) of the agents belonging to our scenario are as follows: REP_1=0.79; REP_2=0.76; REP_3=0.66; REP_4=0.42; REP_5=0.96; REP_6=0.62; REP_7=0.66; REP_8=0.5; REP_9=0.77; REP_{10}=0 and REP_{11}=0.58. At this point, we calculate the trust (see Formula 2). Fixed the agent a_1, we compute the opinion (i.e., trust) that a_1 has with regard to other agents. Recall that this value changes with α. We consider three values of α. In particular, α=1 means that the agent a_1 considers only the opinions of the agents with whom he/she interacted in the past (contrariwise, α=0). Finally, α=0.5 means that the agent a_1 considers in the same way both the opinions of the agents with whom he/she interacted both others. For detail, see Table IV. Let ξ be a threshold fixed by the agent a_1, we suggest only those products that have REC_a^p greater than ξ (in our case, we fix $\xi > 4$). In particular, we note that the agent a_5 that has a high value of reliability for the agent a_1 buys the products p_8 and p_9. Also, a_5 assigns to them a high rate while the rest of the community gives a very low rate. Surely a_1 would be very interested in these products, because he/she trusts a_5 with a high value. At this point, we see how the algorithm behaves. The agent a_1 receives, at the current step, some recommendations by the other agents, in response

userID	productID	categoryID	rating	helpfulness
1	9	3	3	5
1	5	2	3	5
1	6	2	5	2
1	7	3	1	6
1	8	3	5	6
2	1	1	3	5
2	2	1	4	6
2	5	2	4	5
2	6	2	5	5
2	8	3	5	2
3	1	1	4	2
3	8	3	5	3
3	2	2	3	5
3	4	2	6	6
4	1	1	5	1
4	3	1	5	2
4	6	2	3	6
4	9	3	2	1
5	1	1	2	6
5	3	1	2	6
5	4	2	4	5
5	8	3	6	6
5	9	3	6	6
6	6	2	2	6
6	5	2	4	2
6	7	3	1	4
6	8	3	0	3
7	1	1	4	4
7	9	3	4	6
7	5	2	5	3
7	8	3	4	3
8	1	1	5	2
8	3	1	5	0
8	6	2	2	5
8	9	3	3	5
9	2	1	4	4
9	6	2	2	5
9	9	3	3	5
11	1	1	5	3
11	3	1	3	3
11	2	2	4	3
11	9	3	4	5

TABLE II
AN EXAMPLE OF DATABASE

to previous recommendation requests (see Table III). If α=1, we suggest to a_1 the products p_5, p_6, p_8 and p_9. In this case, we consider the recommendations that come from agents that have a high reliability. In fact, these products were acquired and evaluated good by agents a_5 and a_2. Comparing these results with truly user-purchased products, it is visible that four out of four products were actually purchased by a_1. If α=0, we suggest to a_1 the products p_4 and p_5. This is correct because they are products purchased by agents who have a high reputation in the community. But, these agents did not interact directly with a_1 therefore they do not know his/her preferences. Indeed, only the product p_5 is of interest to a_1. If α=0.5, we suggest to a_1 the products p_4, p_5 and p_9 . This choice is a good compromise, since two of the three products are of liking for the agent a_1.

α	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
0	3.74	3.76	3.35	4.82	4.32	3.62	1.00	3.61	3.95
0.5	3.56	3.83	3.17	4.50	4.42	3.95	1.00	3.89	4.17
1	3.28	4.00	2.87	4.00	4.58	4.52	0	4.42	4.49

TABLE III
THE RECOMMENDATIONS TO THE AGENT a_1

u	v	$t_{uv}(\alpha = 0)$	$t_{uv}(\alpha = 0.5)$	$t_{uv}(\alpha = 1)$
1	2	0.76	0.68	0.6
1	3	0.66	0.33	0
1	4	0.42	0.21	0
1	5	0.96	0.98	1
1	6	0.62	0.31	0
1	7	0.66	0.74	0.82
1	8	0.50	0.45	0.4
1	9	0.77	0.54	0.3
1	10	0	0	0
1	11	0.58	0.34	0.1

TABLE IV
SIMULATION PARAMETERS WHEN VARYING α

With the introduction of the cliques in the community, we can consider the assumption made in the Section III. Recall that, for the agents that are in the same cliques, the trust is equal to $t_{x,y} + \Delta$. If $t_{x,y} + \Delta$ is greater than 1, $t_{x,y}$ = 1. This means that for any value of $Delta$, $t_{x,y}$ can not assume a value greater than 1. Now, we can calculate the recommendations (see Equation 3) to the agent a_1 for all the products in the community in presence of the cliques and $\Delta = 0, 5$. The table V shows the results obtained.

α	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
0	3.74	3.76	3.33	4.79	4.42	3.64	1.00	3.65	3.99
0.5	3.92	3.83	3.16	4.47	4.50	3.96	1.00	3.90	4.17
1	3.32	4.00	3.03	4.00	4.62	4.52	0	4.37	4.49

TABLE V
THE RECOMMENDATIONS TO THE AGENT a_1 IN THE PRESENCE OF CLIQUES

The results in the presence of the cliques are best, because a_3 and a_7 know better a_1 and consequently are able to make targeted recommendations.

VI. THE APPLICATION OF OUR PROTOCOL TO THE SOCIAL NETWORK CIAO

In this section, we describe the implementation of our protocol for the social network CIAO. CIAO dataset, described in [33], consists in a matrix with a total of about 36k rows, each of them representing an event in the virtual community, in the form userID, productID, categoryID, rating, helpfulness, timestamp. More in detail, CategoryID represents the commercial category of the product for which the user has released a rating, and the helpfulness (a number between 1 and 6) represents the level of satisfaction of the other users for that rating. In addition, a dataset representing trust relationships is available. It consists in a list of pairs of user IDs (x, y), where each of them represents a trust relationship among user x and user y.

The measure REP (see Formula 1) for each agent is calculated using the code shown in Algorithm 1.

Listing 2. Algorithm 1: computeREP

```
Input rev: the review matrix generated by the agents
Input numAg: the number of agents in the community
Variable j: a real number
Variable x: a real number
Variable contScore: a real number
Variable sum: a real number
Variable repValue: a vector
1:   sum:=0
2:   contScore:=0
```

```
3:  for x:=1 to numAg do
4:    for j:=0 to length of rev do
5:      if x = rev[j][1] then
6:        contScore := contScore + 1
7:        sum := sum + rev[j][4];
8:      end if
9:    end for
10:   if (sum $\neq$ 0) AND (contScore $\neq$ 0) then
11:     repValue[x]:= (sum/contScore)/6;
11:     sum=0;
12:     contScore=0;
13:   end if
14:   else
15:     repValue[x]=0;
16:   end else
17: end for
```

First, for each agent x of the community (Line 3), if the index x corresponds to the one contained in the review matrix (Line 3), the algorithm saves all the helpfulness related to its reviews within the variable sum (Line 7) and increases the variable $contScore$ to keep track of the number of reviews affected by x (Line 6). At this point, if x has issued at least one review (Line 10), the algorithm saves to the x position of the vector $repValue$ the value of REP (Line 11). Otherwise, the rep value is set to zero (Line 15).

The measure $TRUST$ (see Formula 2) for each agent is calculated using the code shown in Algorithm 2.

Listing 3. Algorithm 2: computeTRUST

```
Input alpha: a real number
Input delta: a real number
Input i: a real number
Input m: a real number
Input cliques: an arrayList of arrayList
Input repValue: a vector
Input adjacency_matrix: a SparseMatrix
Input trust_out_file: a file
Variable inter: a HashSet
Variable verify: a real number
Variable delta: a real number
1:  inter:= INTESECT (cliques(i), cliques(m))
2:  if (adjacency_matrix.CONTROL(i, m) = true) AND (lenght
of inter = 0) then
3:      trust(i, m) := alpha + (1-alpha) * repValue[m]
4:  end if
5:  else if (adjacency_matrix.CONTROL(i, m) = false) AND (lenght
of inter = 0) then
6:      trust(i, m) := (1-alpha) * repValue[m]
7:  end else if
8:  else if (adjacency_matrix.CONTROL(i, m) = true) AND (length
of inter $\neq$ 0) then
9:      verify:=alpha + ((1-alpha)*repValue[m])+delta
10:     if (verify <= 1) then
11:        trust(i,m):= verify
12:     end if
13:     else
14:        trust(i,m):= 1
15:     end else
16: end else if
17: else if (adjacency_matrix.CONTROL(i, m) = false) AND (length
of inter $\neq$ 0) then
18:     verify:= ((1-alpha)*repValue[m])+delta;
19:     if verify <= 1 then
20:        trust(i,m):= verify
21:     end if
22:     else
23:        trust(i,m):= 1
24:     end else
25: end else if
26: if (trust(i,m)>0) AND (trust_out_file $\neq$ NULL) then
27:     insert trust(i,m) in trust_out_file
28: end if
```

First, the algorithm verifies if the agents i and m have cliques in common and stores them in $inter$ (Line 1). The algorithm performs some checks to determine how to calculate the trust. If there is a direct link between i and m but they do not have cliques in common (Line 2), the algorithm calculates $TRUST_{i,m} = \alpha + (1-\alpha) \cdot repValue[m]$ (Line 3). If there is no interaction between i and m (Line 5), the algorithm calculates $TRUST_{i,m} = (1-\alpha) \cdot repValue[m]$ (Line 6). If there is a direct link between i and m but they have at least a clique in common (Line 8), the algorithm calculates $TRUST_{i,m} = \alpha + (1-\alpha) \cdot repValue[m] + delta$ and store it in the variable $verify$ (Line 9). The algorithm checks the trust to verify if it takes a value equal to or less than 1 (Lines 10-14). Finally, if there is no direct link between i and m but they have at least a clique in common (Line 17), the algorithm calculates $TRUST_{i,m} = (1-\alpha) \cdot repValue[m] + delta$ and verify if its value is equal to or less than 1 (Lines 18-23). Whenever $TRUST_{i,m} \neq 0$, its value is inserted into an output file (Lines 26-27).

Finally, the measure REC_x^p (see Formula 3) for each agent x is calculated using the code shown in Algorithm 3.

Listing 4. Algorithm 3: computeREC

```
Input rev: the review matrix generated by the agents
Input trust: the trust matrix of the agents
Input numProd: the number of products in the community
Variable recProd: the matrix of recommended products
Variable recAg: a real number
Variable s: a real number
Variable sumRec: a real number
Variable y: a real number
1:  recAg:=1
2:  s:=0
3:  sumRec:=0
4:  for y:=1 to numProd do
5:    for n:=0 to length of rev do
6:      if (y = rev[n][1]) AND (recAg $\neq$ rev[n][0])
AND (trust.CONTROL(recAg, rev[n][0]) = true) then
7:        s:= s + rev[n][3] * trust(recAg, rev[n][0])
8:        sumRep:= sumRep + trust(recAg, rev[n][0])
9:      end if
10:   end for
11:   if (s $\neq$ 0) AND (sumRec $\neq$ 0) then
12:     recProd(recAg, y):= s/sumRec
13:   end if
14:   else
15:     recProd(recAg, y):= 0
16:   end else
17:   s:= 0
18:   sumRec:= 0
19:   recAg:= recAg + 1
17:end for
```

First of all, the algorithm initialises the variables $recAg$, s and $sumRec$ (Lines 1-3). $recAg$ stores the agent's index to which we want to recommend the product. It is incremented at the end of each for loop (Line 19). For each product y of the community and for each row of rev (Lines 4-5), if there is a revision for the product y that has not been inserted by $recAg$ and there is a direct link between the reviser (i.e., $rev[n][0]$) and $recAg$, the algorithm calculates the numerator and the denominator of the Formula 3 (Lines 6-8). Then, the algorithm verifies that these values are different from zero and stores this value in a matrix in row $recAg$ and column y (Lines 11-15). When the for loop on the revision matrix ends, s and $sumRec$ are reset and $recAg$ is increased by 1 (Lines 17-19).

VII. Conclusion

In this paper, we introduce a trust model that integrates reliability and reputation in an OSN organised by cliques. In particular, we considered two important parameters: the relevance given to the reliability with respect to the reputation and the threshold of recommendation under which a product can be considered as not interesting. Furthermore, we present a realistic example and the results show that, combining the opinion of the whole community (reputation) with that of the agents who had direct interactions with her/him (reliability), the algorithm produces good recommendations. In addition, the algorithm is more accurate when the cliques are introduced in the OSN. Finally, we apply our approach to the social network CIAO. Our ongoing research is focused on better analysing the behaviour of this algorithm. In particular, we are theoretically studying the efficiency and the effectiveness of the approach from a statistical viewpoint, and we are performing some experimental campaigns for practically evaluating the advantages introduced by our approach.

References

[1] F. Buccafurri, L. Fotia, and G. Lax, "Allowing non-identifying information disclosure in citizen opinion evaluation," in *International Conference on Electronic Government and the Information Systems Perspective*. Springer, 2013, pp. 241–254.

[2] ——, "Allowing privacy-preserving analysis of social network likes," in *Privacy, Security and Trust (PST), 2013 Eleventh Annual International Conference on*. IEEE, 2013, pp. 36–43.

[3] ——, "Social signature: Signing by tweeting," in *International Conference on Electronic Government and the Information Systems Perspective*. Springer, 2014, pp. 1–14.

[4] P. De Meo, A. Nocera, D. Rosaci, and D. Ursino, "Recommendation of reliable users, social networks and high-quality resources in a social internetworking system," *Ai Communications*, vol. 24, no. 1, pp. 31–50, 2011.

[5] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, and G. M. Sarné, "An evolutionary approach for cloud learning agents in multi-cloud distributed contexts," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2015 IEEE 24th International Conference on*. IEEE, 2015, pp. 99–104.

[6] P. D. Meo, K. Musial-Gabrys, D. Rosaci, G. M. Sarnè, and L. Aroyo, "Using centrality measures to predict helpfulness-based reputation in trust networks," *ACM Transactions on Internet Technology (TOIT)*, vol. 17, no. 1, p. 8, 2017.

[7] P. De Meo, F. Messina, D. Rosaci, and G. M. Sarné, "Recommending users in social networks by integrating local and global reputation," in *International Conference on Internet and Distributed Computing Systems*. Springer, 2014, pp. 437–446.

[8] P. De Meo, L. Fotia, F. Messina, D. Rosaci, and G. M. Sarné, "Forming classes in an e-learning social network scenario," in *International Symposium on Intelligent and Distributed Computing*. Springer, 2016, pp. 173–182.

[9] D. Rosaci, "Finding semantic associations in hierarchically structured groups of web data," *Formal Aspects of Computing*, vol. 27, no. 5-6, pp. 867–884, 2015.

[10] P. De Meo, E. Ferrara, D. Rosaci, and G. M. Sarné, "Trust and compactness in social network groups," *IEEE transactions on cybernetics*, vol. 45, no. 2, pp. 205–216, 2015.

[11] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[12] P. De Meo, F. Messina, D. Rosaci, and G. M. Sarné, "An agent-oriented, trust-aware approach to improve the qos in dynamic grid federations," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 5411–5435, 2015.

[17] W. Sherchan, S. Nepal, and C. Paris, "A survey of trust in social networks," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 47, 2013.

[13] ——, "Improving grid nodes coalitions by using reputation," in *Intelligent Distributed Computing VIII*. Springer, Cham, 2015, pp. 137–146.

[14] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. M. Sarné, "A trust model for competitive cloud federations," *Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 469–474, 2014.

[15] P. De Meo, F. Messina, D. Rosaci, and G. M. Sarné, "Combining trust and skills evaluation to form e-learning classes in online social networks," *Information Sciences*, vol. 405, pp. 107–122, 2017.

[16] F. Messina, G. Pappalardo, D. Rosaci, C. Santoro, and G. Sarné, "A trust-based approach for a competitive cloud/grid computing scenario," *Intelligent Distributed Computing VI*, pp. 129–138, 2013.

[18] E. Majd and V. Balakrishnan, "A trust model for recommender agent systems," *Soft Computing*, pp. 1–17, 2016.

[19] S. Tadelis, "Reputation and feedback systems in online platform markets," *Annual Review of Economics*, vol. 8, no. 1, 2016.

[20] A. Comi, L. Fotia, F. Messina, D. Rosaci, and G. M. Sarné, "A partnership-based approach to improve qos on federated computing infrastructures," *Information Sciences*, vol. 367, pp. 246–258, 2016.

[21] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, and G. M. Sarné, "A reputation-based approach to improve qos in cloud service composition," in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2015 IEEE 24th International Conference on*. IEEE, 2015, pp. 108–113.

[22] D. Rosaci, G. M. Sarné, and S. Garruzzo, "Integrating trust measures in multiagent systems," *International Journal of Intelligent Systems*, vol. 27, no. 1, pp. 1–15, 2012.

[23] L. Xiong and L. Liu, "Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 7, pp. 843–857, 2004.

[24] H. Xia, Z. Jia, L. Ju, X. Li, and Y. Zhu, "A subjective trust management model with multiple decision factors for manet based on ahp and fuzzy logic rules," in *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on*. IEEE, 2011, pp. 124–130.

[25] X. Liu, A. Datta, and E.-P. Lim, *Computational Trust Models and Machine Learning*. CRC Press, 2014.

[26] M. Wessner and H.-R. Pfister, "Group formation in computer-supported collaborative learning," in *Proceedings of the 2001 international ACM SIGGROUP conference on supporting group work*. ACM, 2001, pp. 24–31.

[27] A. Comi, L. Fotia, F. Messina, D. Rosaci, and G. M. Sarné, "Grouptrust: Finding trust-based group structures in social communities," in *International Symposium on Intelligent and Distributed Computing*. Springer, 2016, pp. 143–152.

[28] L. R. Hoffman and N. R. Maier, "Quality and acceptance of problem solutions by members of homogeneous and heterogeneous groups." *The Journal of Abnormal and Social Psychology*, vol. 62, no. 2, p. 401, 1961.

[29] A. Comi, L. Fotia, F. Messina, G. Pappalardo, D. Rosaci, and G. M. Sarné, "Forming homogeneous classes for e-learning in a social network scenario," in *Intelligent Distributed Computing IX*. Springer, 2016, pp. 131–141.

[30] ——, "Using semantic negotiation for ontology enrichment in e-learning multi-agent systems," in *Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on*. IEEE, 2015, pp. 474–479.

[31] S. Basu Roy, L. V. Lakshmanan, and R. Liu, "From group recommendations to group formation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1603–1616.

[32] A. A. Kardan and H. Sadeghi, "An efficacious dynamic mathematical modelling approach for creation of best collaborative groups," *Mathematical and Computer Modelling of Dynamical Systems*, vol. 22, no. 1, pp. 39–53, 2016.

[33] J. Tang, X. Hu, H. Gao, and H. Liu, "Exploiting local and global social context for recommendation." in *IJCAI*, vol. 13, 2013, pp. 2712–2718.