A Trust-Augmented Voting Scheme for Collaborative Privacy Management

Yanjie Sun[1,2], Chenyi Zhang[1,3], Jun Pang[1],
Baptiste Alcade[1]⋆ and Sjouke Mauw[1]

[1] University of Luxembourg, L-1359, Luxembourg
[2] Shandong University, Jinan, 250101, China
[3] University of New South Wales, Sydney 2052, Autralia

Abstract. Social networks have sprung up and become a hot issue of current society. In spite of the fact that these networks provide users with a variety of attractive features, much to users' dismay, however, they are likely to expose users private information (unintentionally).
In this paper, we propose an approach which is intended for addressing the problem of collaboratively deciding privacy policies for, but not limited to, shared photos. Our proposed algorithm utilizes trust relations in social networks and combines it with the Condorcet preferential voting scheme. An optimization is developed to improve its efficiency. Experimental results show that our trust-augmented voting scheme performs well. An inference technique is introduced to infer a best privacy policy for a user based on his voting history.

1 Introduction

Social networking is one of the greatest inventions on the Internet during the last ten years. Social network sites provide users platforms to socialize both in the digital world and in the real world, for making friends, information exchange and retrieval, and entertainment. Some of the largest ones, such as Facebook [1] and MySpace [2], provide services to hundreds of millions of registered users. However, partly due to the intention to attract as many users as possible for their commercial success, social networks tend to intentionally or unintentionally expose private information of existing users. Privacy is becoming an important and crucial research topic in social networks. A number of scholars have studied it from different viewpoints, e.g. [3–7]. Moreover, the excessively expanded number of users also bring difficulties into the management of these sites, so that designing effective mechanisms to coordinate users' opinions over their privacy becomes an emerging issue.

As a shared platform, resources in a social network may be co-owned by a number of users. For instance, documents can be co-authored and several users may appear in a same photo. Such co-ownership might cause breach of privacy.

⋆ Supported by the grant TR-PDR BFR08-038 from the Fonds National de la Recherche, Luxembourg.

For example, suppose user Alice wishes to publish on her personal page a picture which contains Bob's image, this action may cause exposure of Bob's privacy, regardless of Bob's personal will. In response to this issue, most of the social network sites choose to place the burden of privacy setting solely on the owners of the resources, to which we hold a different stance. We believe it might be more desirable to let all co-owners participate in the privacy setting. In this paper, we mainly focus on the particular problem of how to merge privacy opinions from co-owners of shared resources.

Voting is a natural choice to build a mechanism which takes individual's preferences on their privacy policies into a joint decision reflecting the "general will" of the group of people who are sharing a piece of data. Siquicciarini *et al.* [8] propose a game theoretical method based on the Clarke-Tax mechanism [9], which can maximize the social utility function by encouraging truthfulness among people in the group. This induces a nice property that the final decision cannot be manipulated by individuals, as users express their true opinions about the privacy preference. However, their method is not as simple as it is claimed to be, as it requires each user to compute a value for each different preference and the user-input values are essential for their method to derive a joint decision. We argue that this requirement is not realistic and it makes users less interested in participating collaborative privacy control.

Instead, in this paper we propose a different but novel solution, by combining trust in social networks with a well-known preferential voting scheme. The trust relations are inherent in social networks and can be easily derived among users, for example, by comparing user profiles or computing the distance of users in a social network. We believe that trust should play an important role especially when users cooperate to decide the privacy policy on a shared resource. In a preferential voting scheme users are required to give an order of their privacy preferences rather than only select a single choice. This allows users to express their opinions on different privacy policies in a more comprehensive way. Moreover, users are not required to associate values to preferences. Comparing to the method of Siquicciarini *et al.* [8], ours is simple to use. The above discussion reflects the two design rationales *expressiveness* and *simplicity of use* in our mind – these considerations lead us to a method for collaborative privacy control which is as simple as possible without losing expressive power.

The rest of the paper is organized as follows. Sect. 2 introduces notions of trust in social networks and Condorcet's preferential voting scheme. In Sect. 3 we propose a new algorithm that enhances traditional Condorcet's voting scheme by taking the trust relation into account. We also propose a heuristic to improve the performance of this new algorithm. Sect. 4 presents experimental data on comparing the algorithms previously introduced, which also justifies the correctness of our heuristic approach. Sect. 5 provides an inference technique to automatically recommend a default vote for a user based on his profile setting and voting history. We conclude the paper in Sect. 6.

2 Preliminaries

2.1 Trust in Social Networks

Literature shows that social life is simply not possible without trust [10, 11]. In particular, trust relations are central to cooperation, i.e., the social process aiming at the increase or preservation of the partners' power, wealth, etc. Online social networks reflect human social relations in the Internet, allowing users to connect to people they know, to share data (e.g., video, photos, text), and to have group activities (e.g., games, events). A number of social structures have been introduced in social networks, such as friendship, group membership and virtual family. The notion of trust is naturally present in social networks, and moreover, in contrast to real life, it can be quantified and made explicit.

Trust has been defined in several different ways. The definition of trust adopted here, first formulated by Gambetta [12], is often referred to as "reliability trust". Thus, we define *trust* as the belief or subjective probability of the *trustor* that the *trustee* will adequately perform a certain action on which the trustor's welfare depends. Trust is hence a quantifiable relation between two agents. There are mainly two approaches to trust quantification, namely by the evaluation of the similarity, or by an analysis of relevant past events (stored in a history) between two entities.

The similarity between two entities is a distance function that can take various attributes into account, such as *social* (e.g., gender, location, company, etc.), and *behavioral* (e.g., the way one ranks or buys) attributes. Intuitively, the closer the two entities are w.r.t. an attribute (i.e., the distance is small), the more likely the trust relation will be strong. In practice, a social network like Facebook implements a *social similarity* mechanism in order to suggest to the user people that she might consider as friends. For instance, if Alice and Bob both have Clare and Danny in their friend list but Bob also has Elisabeth as a friend, then Elisabeth might be suggested to Alice.

In the second approach, a sequence of past events concerning a specific action can be analyzed to predict the likelihood that the same action will be correctly performed if requested. This analysis can consist in checking a property over the history [13, 14], or the computation of a probability (e.g., Hidden Markov Models [15]). Moreover, trust transitivity can be used when an entity wants to evaluate the trust in an unacquainted entity, e.g., when the trustor does not have access to the trustee's profile, or has never interacted with her. Trust transitivity is defined as the possibility for the trustor to use trust information from other entities in order to infer a trust evaluation towards the trustee, i.e., derive a trust value from a trust graph. In social networks, we can use the trust over friendship relations as transitive relations. Computational models for trust transitivity can be found in the literature [16–19], and can be applied to social networks.

Our main motivation to incorporating trust in collective privacy management is that people's opinion in a social network can be evaluated by taking trust relations into account. For instance, when Alice rates a video that is made by Bob, Clare evaluates this data item through the trust she has assigned to Alice (as a referee) and Bob (as a film-maker). Similarly, combining users' opinions is usually required to decide privacy policies of shared contents, which leads us to a trust-augmented voting scheme. In the following sections, we define trust as a function that assigns a value in $[0, 1]$ to every (ordered) pair of users, and assume that trust values among users can be efficiently computed in social networks by using the approaches as discussed above. The meaning of a value 0 is that a trustor fully distrusts a trustee such that his opinion will be completely disregarded, while 1 means that a trustor fully trusts a trustee.

2.2 Privacy Policies

In this paper, we simply refer to privacy polices as the set of users who are allowed access to shared resources. For instance, for a co-owner picture, the available policies are (*i*) visible only to the owner (P_1), (*ii*) visible only to those tagged, also called co-owners, in the picture (P_2), (*iii*) visible to friends of the tagged users (P_3) and (*iv*) available to everyone (P_4). Throughout the paper, we use $P_1, P_2, \ldots, P_n$ to range over such policies.

2.3 Condorcet's Preferential Voting Scheme

There exist a number of voting systems in the literature. In single candidate systems each vote refers to a single choice, which sometimes may not be able to encode more comprehensive opinions. For example, a voter cannot express that he is initially willing to vote for Alice, but in case that Alice fails to be elected, he will vote for Bob among the rest of the candidates. In such a situation preferential systems can be applied to express more precise and more comprehensive ideas from the voters. In this paper we present a prefential voting scheme which is extended from a system that is originally developed by Condorcet in late eighteenth century. For a complete description of Condorcet's voting system we refer to [20], which also explains why in a certain sense Condorcet's scheme may be regarded as 'optimal'.

In a preferential voting system, a ballot consists of a (preferential) list of all candidates. For every pair of candidates appearing in the list, say C_1 and C_2, their relative positions reflect the voter's preference, e.g., the case that C_1 precedes C_2 in the list indicates that the voter prefers C_1 to C_2. Such a list implies a total order on the set of candidates expressing the complete opinion from the particular voter. A *voting profile* is a collection (or a multi-set) of all the cast ballots.

A voting profile may also be described as a *weighted matrix* of size $|\mathbb{C}| \times |\mathbb{C}|$, where $\mathbb{C}$ is the set of candidates. A cell in a weighted matrix with row C_1 and column C_2 is filled with $w(C_1, C_2)$, which is the number of votes that prefer C_1 to C_2. We further define a *Condorcet directed graph* $G = (V, E, W)$ such that $V = \mathbb{C}$ is the set of vertices, and E is the set of edges, which is defined as the set $\{(C_1, C_2) \in V \times V : w(C_1, C_2) \geq w(C_2, C_1)\}$. The function $W : E \to \mathbb{N}$, determines the labelings of the edges and is defined by $W(C_1, C_2) = w(C_1, C_2) - w(C_2, C_1)$, i.e., the difference between the votes preferring C_1 to C_2 and the votes that prefer C_2 to C_1. At the end of a voting procedure, a voting profile (or equivalently, a weighted matrix) needs to be evaluated in a certain way, in order to get a final result. There is a famous criterion applied in the calculation of a final winner as originally advocated by Condorcet.

Principle 1 *(Condorcet Winner) If there is a candidate that beats every other candidate in one-to-one comparison, that candidate should be the winner.*

Plainly, a Condorcet winner is a vertex in the Condorcet directed graph with out-degree $|\mathbb{C}| - 1$. We adopt Condorcet's method in the following example.

Example 1. Suppose there are three users Alice (C), Bob (B) and Clare (C) tagged in a picture owned by Alice. Alice wants to publish the picture on her personal page in the network, therefore she needs to negotiate with Bob and Clare to reach an agreement on the privacy policy associated to the picture. Suppose the available policies are defined in Sect. 2.2. A (preferential) voting form is made for Alice, Bob and Clare in which they fill in their preferential list on the available policies, as shown in the voting profile on the left of Fig. 1.

Voter \ Policy	P_1	P_2	P_3	P_4
A	1	2	3	4
B	2	3	1	4
C	3	1	2	4

	P_1	P_2	P_3	P_4
P_1	–	2	1	3
P_2	1	–	2	3
P_3	2	1	–	3
P_4	0	0	0	–

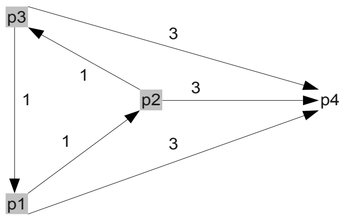

Fig. 1. A Condorcet voting example: the voting profile (left), the weighted matrix (middle), and the Condorcet directed graph (right).

The weighted matrix and the Condorcet directed graph are also sketched in Fig. 1. As one can see, P_4 is the least preferred by all users. However, there exists a *general tie* between policies P_1, P_2 and P_3. This situation is referred to as the *Condorcet paradox*, meaning that the generated Condorcet directed graph is cyclic on its top vertices, and unfortunately, as we will show, the traditional method of Condorcet is unable to break such tie in this example.

Condorcet's method adopts what is known as *maximum likelihood estimation*. A philosophical assumption is that there exists an invisible total ranking, reflecting the true capabilities of all the candidates in an election. In this paper we adopt such mechanism on selecting optimal privacy policies. For every pair of policies P_1 and P_2, such that P_1 procedes P_2 on the (invisible) ranking list, a user is more likely to vote for P_1 than P_2. As Condorcet assumes, each user will choose a better option (P_1 in this case) with some fixed probability p, where $\frac{1}{2} < p < 1$. Taking Example 1, the likelihood of the total order $P_1P_2P_3P_4$ to be the same as the true invisible ranking order, denoted by $L(P_1P_2P_3P_4)$, is calculated by combining the likelihood of every P_i beating P_j with $i < j$ (note that there are six preferential pairs in this case). If "$P_1P_2P_3P_4$" is the true ordering on the candidates, then the chance that we get the current voting profile can be calculated as

$$\begin{aligned} L(P_1P_2P_3P_4) &= L(P_1P_2) \cdot L(P_2P_3) \cdot L(P_1P_3) \cdot L(P_1P_4) \cdot L(P_2P_4) \cdot L(P_3P_4) \\ &= \binom{3}{2}\left(p^2(1-p)^1\right) \cdot \binom{3}{2}\left(p^2(1-p)^1\right) \cdot \binom{3}{1}\left(p^1(1-p)^2\right) \cdot \\ &\quad \binom{3}{3}\left(p^3(1-p)^0\right) \cdot \binom{3}{3}\left(p^3(1-p)^0\right) \cdot \binom{3}{3}\left(p^3(1-p)^0\right) \end{aligned}$$

where $\binom{m}{n} = \frac{m!}{n!(m-n)!}$ for non-negative $m \geq n$. The expression for $L(P_1P_2)$, for example, follows from the fact that 2 out of 3 voters ranked policy P_1 higher than P_2, and thus voted in accordance with the hypothetical true ranking order $L(P_1P_2P_3P_4)$.

It has been pointed out that in practice when comparing the likelihood of two possible orderings, the combinatoric coefficients can be safely ignored [20], given the same voting profile. The only part that needs to be taken into account consists of the exponents over p (note that $\frac{1}{2} < p < 1$). In the case of $L(P_1P_2P_3P_4)$, the power over p is 14. One may also find that the power over p for $L(P_4P_3P_2P_1)$ is 4, thus $P_1P_2P_3P_4$ is more likely to be the true ordering over the privacy policies than $P_4P_3P_2P_1$ by Condorcet's method. As we have mentioned above, in this example we can compute the likelihood for every sequence that is a permutation of the set $\{P_1, P_2, P_3, P_4\}$. In fact, we have $L(P_1P_2P_3P_4) = L(P_2P_3P_1P_4) = L(P_3P_1P_2P_4)$ and it is larger than the likihood of any other sequence. This means that Condorcet's method might select *multiple winners*, as P_1, P_2 and P_3 are all selected in Example 1.

Condorcet voting algorithm. The Condorcet voting algorithm is detailed as in Alg. 1. The algorithm takes a voting profile as input and produces a set of winners as output. Function *getCondorcetWeightedMatrix* translates a voting profile into a weighted matrix, and then in the next step function *getCondorcetDirectedGraph* converts the weighted matrix into a Condorcet directed graph.[4] For example, as shown in Fig. 1, the algorithm translates the voting profile on the left part into the weighted matrix in the middle, and then into the Condorcet directed graph on the right. The rest of the algorithm focuses on how to select a set of top vertices in the Condorcet directed graph. By definition, the set of Condorcet winners are vertices that have an outgoing edge to every other vertex, although in the real world such sets are singletons in most cases. Such set will be returned by function *getWinners*. If *getWinners* returns an empty set, i.e., no Condorcet winners exist, the algorithm will compute the likelihood of all possible sequences and maintain the set of those with the maximal likelihood, and return their first elements as winners.

[4] We represent directed graphs as two-dimensional arrays. For example, if there is a directed edge from i to j with weight $n \geq 0$, then $cdg[i][j]$ has value n, and $cdg[j][k] = -1$ means that there is no edge from j to k.

Algorithm 1 The Condorcet voting algorithm.

```
input : $votingprofile$ : VotingProfile;
output : $winners$ : set $\langle string \rangle$;
var $cwm$      : int[ ][ ]              init  null
    $cdg$      : int[ ][ ]              init  null
    $tlv$      : int[ ]                 init  null
    $sql$      : int                    init  0
    $ml$       : int                    init  0
    $ms$       : set $\langle string \rangle$  init  $\emptyset$

begin
$cwm$ := getCondorcetWeightedMatrix($votingprofile$);
$cdg$ := getCondorcetDirectedGraph($cwm$);
$winners$ := getWinners($cdg$);
if $winners = \emptyset$ then
  $tlv$ := findTopLevelVertices($cdg$);
  for each sequence $sq$ which is a permutation of $tlv$ do
    $sql$ := computeSequenceLikelihood($sq$, $cwm$);
    if $sql > ml$ then
      $ml := sql$;
      $ms := \{sq\}$;
    else if $sql = ml$ then
      $ms := ms \cup \{sq\}$;
    end if
  end for
  $winners$ := getFirstElements($ms$);
end if
end
```

In the above algorithm, not all sequences are required to be involved in the comparison of likelihoods. As a Condorcet directed graph imposes a topological order, the top level vertices compose a subgraph which is a strongly connected component (SCC) in the original graph. Function *findTopLevelSCC* returns the set of vertices that form the SCC. One may easily find that the set of winners can only come from the SCC, thus we only need to compute the sequences initialized by permutations of vertices in the SCC.[5] As in Example 1, only six three-element sequences need to be taken into account: $P_1P_2P_3$, $P_1P_3P_2$, $P_2P_1P_3$, $P_2P_3P_1$, $P_3P_1P_2$ and $P_3P_2P_1$, as P_4 can only be the least preferred. The algorithm will pick up three (total) sequences, $P_1P_2P_3P_4$, $P_2P_3P_1P_4$ and $P_3P_1P_2P_4$, which are with the most likelihood, and produce the winner set $\{P_1, P_2, P_3\}$ by taking the first elements (by the function *getFirstElements*). The restriction to the top-level SCC effectively narrows the range of winners we need to consider, which greatly improves the performance of the selection procedure. It is easy to see that the

[5] There is no need to compute a whole sequence containing elements not in the SCC, as Condorcet's methods is *locally stable* [20], the particular order of less preferred candidates would not influence the final voting result.

running time of the algorithm has an upper bound in $O(|V|!)$, due to checking the likelihood of all possible sequences of nodes in the SCC.

3 A Trust-Augmented Voting Scheme

3.1 Incorporating Trust as Weighted Votes

In some situations not all voters are equal. Typical examples include decision-makings in a shareholder's meeting where the weight of each voter corresponds to his volume of share. Likewise in social networks, users' opinions on deciding a privacy policy do not necessarily carry the same weight. For example, Alice has a picture in which there are Bob, Clare, Danny and Elisabeth, and she wants to publish that picture in her album. She is willing to give right to the co-owners of the picture, i.e. Bob, Clare, Danny and Elisabeth, on deciding whether the privacy level of that picture is P_1 or P_2 (see their definitions in Sect. 2.2). Here we suppose that Bob is a friend of Alice and Clare is a friend of Bob but not a direct friend of Alice, i.e., Clare is a friend of a friend of Alice. Similarly, Danny is a friend of Alice and Elisabeth is a friend of a friend of Alice. In this case it seems more reasonable to give Bob's and Danny's opinion more weight than Clare's and Elisabeth's. In this section we propose an extension of Condorcet's voting system for weighted votes. The weight of each vote reflects the trust level of the owner of the shared resource having on the co-owners in their votes for setting a privacy policy for publishing the resource.

Voter \ Policy	P_1	P_2
A	1	2
B	2	1
C	1	2
D	2	1
E	1	2

Voter(trust) \ Policy	P_1	P_2
A (1.0)	1	2
B (0.9)	2	1
C (0.2)	1	2
D (0.8)	2	1
E (0.3)	1	2

Fig. 2. An example: The effect of trust in collaborative privacy management.

We sketch the voting results from this scenario in Fig. 2, and it is easy to find that P_1 is the winner according to the left table, as it receives three votes while P_2 only receives two. However, if weights (interpreted as the trust level of Alice in the co-owners to make the right decision on privacy preferences) are associated to votes, then P_2 is the winner, as it is supported by 1.7 weighted votes while P_1 supported only by 1.5. This example clearly shows that trust relations in social networks, if carefully incorporated into decision making procedures, can affect the results in collaborative privacy management.

In the original Condorcet voting system all votes carry the same weight, and the preferential orders can be compared by only looking at the (integer) exponents of p regarding to their likelihoods. In this paper we measure the likelihoods of these orders by allowing discounted votes to reflect the degree of trust of a user by the owner of a resource, so that each vote carries a real valued weight in $[0,1]$ instead of always being an integer 1.

Algorithm 2 A trust-augmented Condorcet voting algorithm.

```
input : $trustvotingprofile$ : VotingProfile;
output : $winners$ : set $\langle string \rangle$;
var $cwm$    : double[ ][ ]      init  null
    $cdg$    : int[ ][ ]         init  null
    $tlv$    : int[ ]            init  null
    $sql$    : double            init  0.0
    $ml$     : double            init  0.0
    $ms$     : set $\langle string \rangle$   init  $\emptyset$

begin
$cwm$ := getCondorcetWeightedMatrix($trustvotingprofile$);
$cdg$ := getCondorcetDirectedGraph($cwm$);
$winners$ := getWinners($cdg$);
(* the rest is the same as Alg. 1 *)
end
```

Trust-augmented Condorcet voting algorithm. The trust-augmented Condorcet voting algorithm is detailed as in Alg. 2. The whole procedure of calculation is exactly the same as that of Alg. 1, except that now the sequence likelihood (sql) and maximal likelihood (ml) are of real valued type instead of integer type, as shown above. A trust-based voting profile, which includes the preference lists and trust level for each participant, is taken as input by the algorithm (e.g., left part of Fig. 3), while the output, a set of winders, remains unchanged. In Fig. 3, we assume that A, as the owner of the picture, fully trusts himself, i.e., his trust value is 1.0. The trust of A in other participants (0.8 for B and 0.6 for C) is also shown in the table. From the trust-based voting profile, the revised Condorcet weighted matrix is obtained (e.g., middle part of Fig. 3). The weighted matrix is slightly different from that in Example 1 in the way that simply counted (integer) votes are replaced by accumulated trust values throughout the table. From the Condorcet directed graph in the right part of Fig. 3, we can find a unique winner P_1, after computing the likelihood of all sequences of nodes in the top level SCC (containing P_1, P_2 and P_3). This can be easily verified since the likelihood of the sequence $P_1P_2P_3P_4$, as calculated as follows, is greater than the likelihood of every other sequence. Note that here we replace the number of votes as integer exponents over "p" and "$1-p$" by their corresponding sums of trust values.

$$\begin{aligned} L(P_1P_2P_3P_4) &= L(P_1P_2)\cdot L(P_2P_3)\cdot L(P_1P_3)\cdot L(P_1P_4)\cdot L(P_2P_4)\cdot L(P_3P_4) \\ &= \binom{3}{2}\left(p^{1.8}(1-p)^{0.6}\right)\cdot\binom{3}{2}\left(p^{1.6}(1-p)^{0.8}\right)\cdot\binom{3}{1}\left(p^{1.0}(1-p)^{1.4}\right)\cdot \\ &\quad \binom{3}{3}\left(p^{2.4}(1-p)^{0}\right)\cdot\binom{3}{3}\left(p^{2.4}(1-p)^{0}\right)\cdot\binom{3}{3}\left(p^{2.4}(1-p)^{0}\right) \end{aligned}$$

It is easy to see that this algorithm has the same time complexity upper bound $O(|V|!)$ as Alg. 1.

Voter\policy	P_1	P_2	P_3	P_4
A (1.0)	1	2	3	4
B (0.8)	2	3	1	4
C (0.6)	3	1	2	4

	P_1	P_2	P_3	P_4
P_1	–	1.8	1.0	2.4
P_2	0.6	–	1.6	2.4
P_3	1.4	0.8	–	2.4
P_4	0.0	0.0	0.0	–

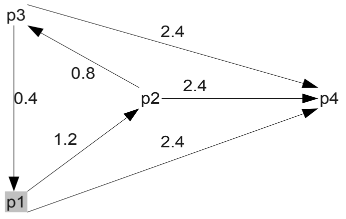

Fig. 3. A trust-augmented Condorcet voting example: the voting profile (left), the weighted matrix (middle), and the Condorcet directed graph (right).

As we have seen so far, the trust level indeed has an impact on collaborative privacy management. Moreover adding trust makes the algorithm better adopted in social networks, since in the real life, advices from different friends affect one's decision differently, depending on the social relationship between the person and his friends. Introducing a trust relation also expands the value space to avoid clashes, as we are going to show experimentally later. In Alg. 2 it is less likely to have unsolved cases as well as multiple winner cases than in Alg. 1. Similar to Alg. 1, the algorithm always selects Condorcet winners whenever they exist. However, in order to decide winners, both Alg. 1 and Alg. 2 may require a calculation of likelihoods for all permutations from the top-level SCC, which potentially takes running time exponential to the number of policies. This fact leads us to the search of more applicable algorithms.

3.2 A Heuristic Algorithm

The algorithm presented in this section provides another way to resolve general ties in the top-level SCC, as well as to reduce the computational time. A comparison between all the algorithms is conducted in Sect. 4.

Taking a Condorcet directed graph, first we have the following arguments. Suppose $w(P_i, P_j)$ is close to 0, it is very likely to be the case that the voters are relatively indifferent with respect to the two policies P_i and P_j. Therefore, regarding to Condorcet's assumption, P_i does not have a significant chance to precede P_j in the underlying invisible order. This justifies our choice in the following algorithm to weaken such difference by adding another (reversing) edge in the graph from P_j to P_i. By doing this, we *equalize* the votes between policies P_i and P_j. Technically, we only apply this operation within the top-level SCC, gradually by starting from the pairs (P_i, P_j) with least $w(P_i, P_j)$, then the pairs with second least weight, and so on. Each time we add new edges it is required to check whether a set of Condorcet winners have been generated in the new graph. The running time of the algorithm has an upper bound of

$O(|V|^2)$, where V is the set of vertices of the Condorcet directed graph, which is much faster than Alg. 2. Nevertheless, the experimental results in Sect. 4 reveal strong similarity with respect to the results of Alg. 3 and Alg. 2, which provides a concrete support to the applicability of Alg. 3.

Algorithm 3 Optimized trust-augmented Condorcet voting algorithm.

```
input : trustvotingprofile: VotingProfile;
output : winners : set ⟨string⟩;
var cwm     : int[ ][ ]      init  null
    cdg     : int[ ][ ]      init  null
    tls     : int[ ][ ]      init  null
    lwes    : set ⟨string⟩   init  $\emptyset$
    nodes   : set ⟨string⟩   init  $\emptyset$

begin
cwm := getCondorcetWeightedMmatrix(trustvotingprofile);
cdg := getCondorcetDirectedGraph(cwm);
winners := getWinners(cdg);
if winners = $\emptyset$ then
  tls := findTopLevelSCC(cdg);
  while true do
    lwes := findLowestWeightEdges(tls);
    addReverseEdges(lwes, cdg);
    nodes := findNodeN-1OutDegree(cdg);
    if nodes ! = $\emptyset$ then
      winners := nodes;
      return
    end if
  end while
end if
end
```

Optimized trust-augmented Condorcet voting algorithm. Similar to Alg. 2, Alg. 3 takes a voting profile as input and produces a set of winning privacy policies. The first part of the algorithm is exactly the same as in the above two algorithms. However, if Alg. 3 cannot find Condorcet winners, it will extract the whole top-level SCC into a subgraph tls. Then starting from the lowest weighted edges, it adds reverse edges into the original Condorcet directed graph cdg, and searches for Condorcet winners in the modified graph. This will repeat until Condorcet winners are found in cdg. The algorithm is guaranteed to terminate before every pair of vertices in the top-level SCC has two connecting edges pointing to each other. Therefore it is bounded by $O(|V|^2)$ where V is the set of vertices in cdg. An application of Alg. 3 on Example 1 with additional trust values (as data shown in Fig. 3) has been depicted in Fig. 4.

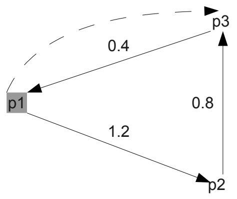

Fig. 4. An example: adding a reverse edge.

4 Experimental Results

We have implemented a program to test the three algorithms. In our test, the voting profiles and the trust levels are randomly generated.[6] The experimental results are obtained from 10,000 cases. We set the policy number in the range of $[3, 10]$, while the number of voters ranging in $[3, 100]$. The optimized algorithm (Alg. 3) can perform well even in the case of a large number of policies due to its improved time complexity.

From Fig. 5, we can find that the number of cases where we cannot find a Condorcet winner in Alg. 2 and Alg. 3 is much less (12%) than that in Alg. 1. This is due to the incorporation of trust. Based on this we can conclude that trust can have a big impact on the voting results. It is also clear that the cases with multiple winners as output is on a steady decrease (15%), which means that the adoption of a trust relation, to a large extent, can effectively increase the possibility of having a unique winner. Moreover, we only see a slight increase in the number of cases with multiple winners for Alg. 3 compared to Alg. 2. Besides, we measured the similarity among the outputs of Alg. 3 and Alg. 2. In 9,519 out of 10,000 cases (i.e., $> 95\%$) they produce the same results. Since theoretically Alg. 2 always generates the best privacy policy, we can conclude that our optimization (Alg. 3) can also produce the best privacy policy for most cases in practice.

Next, we have built a different set of experiments to compare the performance of Alg. 2 and Alg. 3. For each number of policies $[5, \ldots, 10]$, we test the two algorithms on 1,000 randomly generated voting profiles. We calculated the average CPU time for each algorithm to produce an output. From Table 1, it is clear that our optimized algorithm greatly improves the efficiency of Alg. 2 – the performance of Alg. 2 degrades largely when the number of policies increases. All experiments are conducted on a Dell laptop with Intel Core(TM) 2 Duo CPU (2.26GHz) and 1.95GB of RAM.

5 Inference of Privacy Policies

When setting collective privacy management issues among users, each user is required to feed input into the social network system, as complete voting profiles

[6] Alg. 1 does not take the trust levels into account.

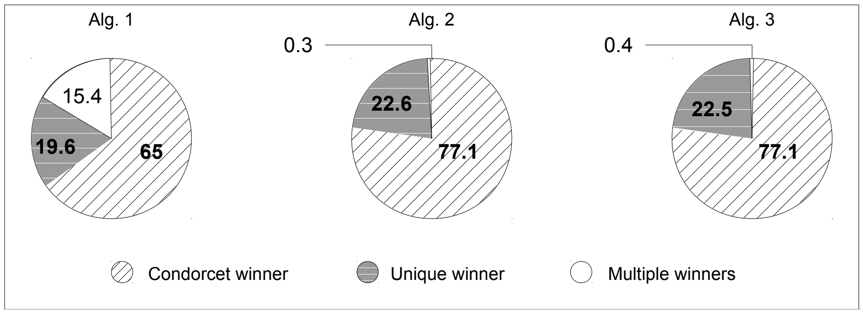

Fig. 5. Case analysis on outputs of the algorithms

Algorithm/#Policy	5	6	7	8	9	10
Alg. 2	100	304	2,289	54,361	334,260	7,597,046
Alg. 3	75	77	80	82	84	91

Table 1. Average CPU time consumption (ms) w.r.t. policy numbers.

are necessary for the algorithms. This may quickly become a cumbersome job of users over time, as the number of co-owned resources increases rapidly. In this section, we introduce an inference technique to automatically suggest suitable preferential votes to the users, which thus relieves the burden of tedious manual inputs. This algorithm only provides a *default vote* for a user, and if he disagrees, it is possible for the user to specify his preferred vote.

In social networks there always exists an initial privacy setting as specified by each user. The proposed inference technique compares a user's initial setting with his history votes, trying to guess his future votes by picking up a vote from the history with the most similarity to the initial privacy setting. Let a user profile's privacy preferential list be a vector $\overrightarrow{P} = \langle p_1, p_2, \ldots, p_n \rangle$, and the privacy preference of a co-owned resource be $\overrightarrow{C} = \langle c_1, c_2, \ldots, c_n \rangle$. Our inference technique picks up an element from a collection of preferences $\{\overrightarrow{C_1}, \ldots, \overrightarrow{C_m}\}$ that has the most similarity to $\overrightarrow{P}$. First, we calculate the cosine similarity [21] between the privacy preference $\overrightarrow{P}$ of a user's profile setting and every $\overrightarrow{C_i}$ in his history votes

$$s_i := cos(\overrightarrow{P}, \overrightarrow{C_i}) = \frac{\sum_{k=1}^{n} p_k \cdot c_{ik}}{\sqrt{(\sum_{k=1}^{n} p_k^2) \cdot (\sum_{k=1}^{n} c_{ik}^2)}}$$

We use $\overline{s}$ to denote the average similarity $\overline{s} = \frac{1}{m}\sum_{k=1}^{m} s_i$. Then the default vote is set as $\overrightarrow{C_i}$ with $|\, s_i - \overline{s}\,|$ being the smallest.

Example 2. Suppose there are three users, Alice (A), Bob (B) and Clare (C), each of which has three co-owned pictures. Their profile settings and history decisions are shown in Fig. 6. Here we apply the inference technique on user

Alice. The cosine similarity between $\overrightarrow{P}(\langle 1,2,3,4\rangle)$ and A's historical decisions $\overrightarrow{C_1}$ $(\langle 2,4,3,1\rangle)$, $\overrightarrow{C_2}$ $(\langle 1,2,3,4\rangle)$ and $\overrightarrow{C_3}$ $(\langle 3,1,4,2\rangle)$ can be calculated as 23/30, 1 and 25/30, respectively. The average similarity is 26/30, to which $\overrightarrow{C_3}$ is the closest and thus is recommended to Alice. We can perform a similar calculation for Bob and Clare.

Voter/Preference	$\boldsymbol{P}$	$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{C}_3$
A	$\langle 1,2,3,4\rangle$	$\langle 2,4,3,1\rangle$	$\langle 1,2,3,4\rangle$	$\langle \mathbf{3},\mathbf{1},\mathbf{4},\mathbf{2}\rangle$
B	$\langle 2,3,1,4\rangle$	$\langle 1,3,2,4\rangle$	$\langle \mathbf{4},\mathbf{1},\mathbf{2},\mathbf{3}\rangle$	$\langle 2,4,3,1\rangle$
C	$\langle 4,3,2,1\rangle$	$\langle \mathbf{4},\mathbf{2},\mathbf{1},\mathbf{3}\rangle$	$\langle 3,4,1,2\rangle$	$\langle 1,2,3,4\rangle$

Fig. 6. An example of the inference of policy preferences.

6 Discussion and Conclusion

Privacy in social networks is a rather complicated issue [22], it has many faces and emerges as a hot research issue in different areas like economics, social science, computer science, and law. In this paper, we have proposed a trust-augmented voting algorithm to solve the particular problem of collective privacy management for shared contents in social networks. Our main idea is to incorporate trust relations among users in social networks as vote weights in the Condorcet preferential voting algorithm. The motivation comes from the facts that trust is naturally inherent in social networks and that a preferential voting scheme is an expressive but simple way for users to formulate their privacy concerns on shared contents. To make the algorithm both efficient and effective, we have developed an optimization for the algorithm to deal with the case when the number of privacy policies is large. An inference technique is used to relieve the users from the burden of manually inputing their privacy preference for each picture.

The algorithms in this paper have been developed mainly from a technical point of view and one can reasonably argue that voting is not the ideal approach to collective privacy management. In the end, the owner of a picture will have to decide for herself whether and how she wants to publish a picture, possibly taking into account the interests of concerned people. We believe that heuristic and algorithmic support to this process will result in a more transparent and hopefully more fair decision process. In the future, we want to build an application as a proof-of-concept for our proposal.

References

1. Facebook: (`www.facebook.com`)
2. MySpace: (`www.myspace.com`)

3. Gross, R., Acquisti, A., John Heinz III, H.: Information revelation and privacy in online social networks. In: Proc. 2005 ACM Workshop on Privacy in the Electronic Society, ACM (2005) 71–80
4. Ellison, N.B., Steinfield, C., Lampe, C.: Benefits of Facebook "Friends": Social capital and college students' use of online social network sites. Journal of Computer Mediated Communication-Electronic **12**(4) (2007)
5. Rosenblum, D.: What anyone can know: The privacy risks of social networking sites. IEEE Security and Privacy **5**(3) (2007) 40–49
6. Carminati, B., Ferrari, E.: Privacy-aware collaborative access control in web-based social networks. In: Proc. 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security. Volume 5094 of LNCS., Springer (2008) 81–96
7. Grossklags, J., Christin, N., Chuang, J.: Secure or insure?: a game-theoretic analysis of information security games. In: Proc. 17th Conference on World Wide Web, ACM (2008) 209–218
8. Squicciarini, A.C., Shehab, M., Paci, F.: Collective privacy management in social networks. In: Proc. 18th International Conference on World Wide Web, ACM (2009) 521–530
9. Clarke, E.H.: Multipart pricing of public goods. Public Choice **11** (1971) 17–33
10. Good, D.: Individuals, interpersonal relations, and trust. In Gambetta, D., ed.: Trust: Making and breaking cooperative relations. Department of Sociology, University of Oxford (1988)
11. Luhmann, N.: Trust and Power. John Wiley and Sons Inc (1979)
12. Gambetta, D., ed.: Trust: Making and breaking cooperative relations. Department of Sociology, University of Oxford (1988)
13. Krukow, K., Nielsen, M., Sassone, V.: A logical framework for history-based access control and reputation systems. Journal of Computer Security **16**(1) (2008) 63–101
14. Eilers, F., Nestmann, U.: Deriving trut from experience. In: Proc. 6th International Workshop on Formal Aspects in Security and Trust. Volume 5983 of LNCS., Springer (2009) 36–50
15. ElSalamouny, E., Sassone, V., Nielsen, M.: HMM-based trust model. In: Proc. 6th International Workshop on Formal Aspects in Security and Trust. Volume 5983 of LNCS., Springer (2009) 21–35
16. Gray, E., Seigneur, J.M., Chen, Y., Jensen, C.: Trust propagation in small worlds. In: Proc. 1st International Conference on Trust Management. Volume 2692 of LNCS., Springer (2003) 239–254
17. Jøsang, A., Marsh, S., Pope, S.: Exploring different types of trust propagation. In: Proc. 4th International Conference on Trust Management. Volume 3986 of LNCS., Springer (2006) 179–192
18. Dong, C., Russello, G., Dulay, N.: Trust transfer in distributed systems. In: Proc. Joint iTrust and PST Conferences on Privacy, Trust Management and Security. Volume 238 of IFIP., Springer (2007) 17–30
19. Alcalde, B., Mauw, S.: An algebra for trust dilution and trust fusion. In: Proc. 6th International Workshop on Formal Aspects in Security and Trust. Volume 5983 of LNCS., Springer (2009) 4–20
20. Young, H.P.: Condorcet's theory of voting. The American Political Science Review **82**(4) (1988) 1231–1244
21. Tan, P.B., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley (2005)
22. Bonneau, J., Preibusch, S.: The privacy jungle: On the market for privacy in social networks. In: Proc. 8th Workshop on the Economics of Information Security. (2009)