Privacy Policy Negotiation in Social Media

JOSE M. SUCH, Lancaster University
MICHAEL ROVATSOS, University of Edinburgh

Social Media involve many shared items, such as photos, which may concern more than one user. The challenge is that users' individual privacy preferences for the same item may conflict, so an approach that just merges in some way the users' privacy preferences may provide unsatisfactory results. Previous proposals to deal with the problem were either time-consuming or did not consider compromises to solve these conflicts (e.g., by considering unilaterally-imposed approaches only). We propose a negotiation mechanism for users to agree on a compromise for the conflicts found. The second challenge we address in this paper relates to the exponential complexity of such a negotiation mechanism. To address this, we propose heuristics that reduce the complexity of the negotiation mechanism and show how substantial benefits can be derived from the use of these heuristics through extensive experimental evaluation that compares the performance of the negotiation mechanism with and without these heuristics. Moreover, we show that one such heuristic makes the negotiation mechanism produce results fast enough to be used in actual Social Media infrastructures with near-optimal results.

Categories and Subject Descriptors: K.4.1 [**Computers and Society**]: Privacy

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Social Media, Privacy, Conflicts, Intimacy, Social Networking Services, Online Social Networks

ACM Reference Format:
Jose M. Such, Michael Rovatsos, 2015. Privacy Negotiation in Social Media. *ACM Trans. Autonom. Adapt. Syst.* 0, 0, Article 0 (0), 29 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Despite the unquestionable success of social media (Facebook recently achieved 1 billion users), privacy is still one of the major concerns with regards to these technologies [Gross and Acquisti 2005]. Moreover, this concern has even been increasing over the last few years because users are more aware of the privacy threats that social media entail [Stutzman et al. 2013]. Most social media users consistently criticise mainstream social media for providing very complex privacy controls. These are often too difficult to understand, require time-consuming manual configuration, and do not allow for appropriate privacy management. Users are required to set many privacy controls (Facebook has 61 privacy controls [Bonneau and Preibusch 2010]), they need to consider a huge space of possible accessors (the average Facebook user has more

Author's addresses: Jose M. Such, School of Computing and Communications, Infolab21, Lancaster University, Lancaster, LA1 4WA, UK. E: j.such@lancaster.ac.uk
Michael Rovatsos, School of Informatics, The University of Edinburgh, Informatics Forum 2.12, 10 Crichton Street Edinburgh EH8 9AB, UK. E: mrovatso@inf.ed.ac.uk

DOI:http://dx.doi.org/10.1145/0000000.0000000

than 200 friends[1]), and they may have to perform fine-grained modifications for many items (the average Facebook user uploads 217 photos[2]). This makes most users unable to cope with the complexity of privacy management in social media, which has led to numerous incidents in which people have lost their jobs, have been cyberbullied, or have lost court cases due to the inappropriate communication of personal information through social media. Empirical evidence shows that this significantly discourages users to either join social media or to show high engagement when they join [Staddon et al. 2012], in terms of how much they participate in social media sites, e.g., the amount of photos they upload, the number of comments they post, etc. Indeed, the most common case is the latter: people usually join social media because they do not want to be left apart, but after they join they do not participate much because they are not able to manage their privacy in a satisfactory way.

To address this problem, new access control paradigms for social media have been recently proposed, such as relationship-based access control [Carminati et al. 2009; Fong 2011; Wang et al. 2012; Such et al. 2012]. These new access control paradigms are aimed at better capturing the nature of information sharing in social media by considering users' relationships as a central concept. This is supported by many studies that provide evidence that user relationships are the main factor that drives human disclosure of personal information [Houghton and Joinson 2010; Wiese et al. 2011; Duck 2007; Strahilevitz 2005], and that they should play a crucial role when defining access control mechanisms for social media [Gates 2007]. In particular, privacy policies in relationship-based access control are neither defined based on individual persons nor on their roles, but on the relationships — and specifically the strength of the relationships or intimacy — that a user has to other users.

The main limitation of state-of-the-art relationship-based access control models is that they only support single user decisions [Such et al. 2014]. That is, these proposals assume that only one user takes the decision of whether or not to grant access to an item. This user is usually the one who uploads the item or shares it in some other way. However, what should we do when the definition of a privacy policy involves more than one user? This is an issue that arises frequently — e.g., when photos depict different people, so that the individual privacy preferences of all of them should be respected when deciding who should be able to view them. The problem is that individual privacy preferences may conflict, as the people in the photo may want to share it with different audiences.

Empirical evidence [Wisniewski et al. 2012] shows that users do actually negotiate among themselves to achieve a *compromise* to solve these conflicts. However, they are forced to do that manually (e.g., phone calls, SMS, etc.) for each and every item and for each and every conflict, which becomes an unmanageable burden on them because of the large number of possible shared items [Thomas et al. 2010] and the large number of possible accessors to be considered by users [Quercia et al. 2012]. Few previous works have proposed mechanisms to deal with this problem such as [Wishart et al. 2010; Squicciarini et al. 2009; Carminati and Ferrari 2011; Hu et al. 2012], but these are either time-consuming or do not allow compromises to solve these conflicts — e.g., the one that uploads the item *unilaterally* imposes a privacy policy or a way to achieve a solution to the conflicts, which leads to unsatisfactory or unacceptable solutions for the other users involved.

Automated negotiation has been satisfactorily used in many other domains to help users reach compromises to resolve conflicts [Lopes et al. 2008]. In this paper, we propose the first automated method to detect conflicts in relationship-based privacy poli-

[1]http://www.theguardian.com/news/datablog/2014/feb/04/facebook-in-numbers-statistics
[2]http://internet.org/efficiencypaper

cies and resolve them using a negotiation mechanism to find adequate compromises. The preferences that determine negotiation behaviour are based on the strength of the relationships among users. As proven by recent experiments [Wiese et al. 2011], and in line with state-of-the-art relationship-based access control mechanisms [Carminati et al. 2009; Fong 2011; Wang et al. 2012], this is the most important factor that users consider when deciding what information to disclose. Moreover, our mechanism uses the well-studied *one-step negotiation protocol* [Rosenschein and Zlotkin 1994] and strategies which are known to be complete, efficient and stable.

The second main challenge we address in this paper relates to the inherent complexity of considering the space of all possible deals users may achieve to solve the conflicts so that the results are optimal. In particular, we show in this paper that this space grows exponentially in the number of conflicts that need to be negotiated for. This is very important because this could prevent the mechanism from being used in actual Social Media infrastructures, as it would be too slow for users to be able to run the mechanism in real-time when they are posting items in the particular Social Media infrastructure.

We overcome the complexity problem by developing a number of suitable heuristics. The aim is to reduce the space of all possible deals users may achieve to only those *most promising*, so that the complexity is reduced while the outputs of the negotiation mechanism remain near-optimal. Through an extensive experimental comparison of the performance of the negotiation mechanism with and without heuristics, we show that they provide a significant search space reduction while remaining near-optimal. One particular heuristic is able to produce results very close to the optimal fast enough to be used in real-world social media.

The remainder of the paper is structured as follows. Section 2 introduces the concept of intimacy and policies for relationship-based access control. Section 3 provides a brief overview of the mechanism. Section 4 describes the method used to detect conflicts. Section 5 proposes the model for ranking possible negotiation outcomes based on existing empirical evidence. Section 6 describes the negotiation mechanism and its complexity. Section 7 introduces the heuristics we propose to reduce the complexity of the problem. Section 8 presents the experiments we conducted and discusses the results obtained. Section 9 reviews the related literature. Finally, Section 10 presents some concluding remarks and describes possible avenues for future work.

2. BACKGROUND

We consider a set of agents $Ag = N \cup T$, where a pair of *negotiating* agents $N = \{a, b\}$ negotiate whether they should grant a set of *target* agents $T = \{i_1, \ldots, i_n\}$ access to a particular item it. For simplicity and without loss of generality, we will consider only a negotiation for one item throughout this paper – for example, a photo that depicts the two users which agents a and b are representing – and hence, we do not include any additional notation for the item in question. The problem we are considering is how a and b can detect whether their individual privacy preferences for the item are conflicting[3], and if they are conflicting, how a and b can achieve an agreement on which agents in T should be granted access to this item.

Negotiating agents have the individual privacy preferences of their users about the item — i.e., to whom of their online friends users would like to share the item if they were to decide it unilaterally. In this paper, we use relationship-based access control

[3]Note that we focus on detecting conflicts once we know the parties that co-own an item and have their individual privacy preferences for the item. We are, however, not proposing a method to automatically detect which items are co-owned and by whom they are co-owned. This is a different problem that is out of the scope of this paper.

[Carminati et al. 2009; Fong 2011; Wang et al. 2012] because this type of access control has emerged as an appropriate way to capture the nature of individual sharing preferences in Social Media, and because it makes the link between individual preferences and our proposed mechanism more intuitive. However, other existing approaches to access control in Social Media — such as the group-based access control models of mainstream Social Media infrastructures (like Facebook lists or Google+ circles), or (semi-)automated approaches like [Fang and LeFevre 2010; Squicciarini et al. 2011; Danezis 2009] — can also be used in conjunction with our proposed mechanism, as we will be pointing out at different points in this paper[4].

2.1. Intimacy

Relationship-based access control models how agents' preferences about disclosure are formed based on the concept of *relationship strength* (or *intimacy*[5]) between two persons [Granovetter 1973]. This is because in the domain of social networks, there is strong evidence that individuals' willingness to share a particular item with another individual is related to how close/intimate their relationship is [Green et al. 2006; Strahilevitz 2005; Houghton and Joinson 2010; Wiese et al. 2011]. It is important to note that intimacy does not equal social distance, which is usually measured as the number of hops (friends) between two users that are not necessarily directly connected with each other. Instead, intimacy is the measure of the relationship strength between two directly connected friends. Intimacy can also be transitive under certain conditions [White and Houseman 2002], so some particular non-directly connected friends may have non-zero intimacy. The rest of non-directly connected friends who do not meet the conditions to have a transitive intimacy would have no intimacy at all — e.g., they would have an intimacy value of 0.

Intimacies in Social Media can be accurately estimated from content users have previously published, using tools that obtain intimacies automatically for Facebook [Gilbert and Karahalios 2009; Fogués et al. 2014]; Twitter [Gilbert 2012]; and the like. These tools consider variables like the time elapsed since the last communication between two users, links shared, number of private messages exchanges, photos together, and many others — please refer to any of these tools to know more about how they work. Even if these tools are not used, users can be asked to self-report their intimacies to their friends, but this would obviously mean more burden on the users. We formally represent the intimacy of two agents as follows:

Definition 2.1. Given two agents $a, b \in Ag$, and a maximum integer intimacy value $\mathcal{Y}$, the *intimacy* between a and b is given as $int(a, b)$, where $int : Ag \times Ag \rightarrow [0, \mathcal{Y}]$.

The maximum possible intimacy $\mathcal{Y}$ depends on the scale used by the particular methods/tools used to obtain intimacy. For example, in Fogués et al. [Fogués et al. 2014] $\mathcal{Y} = 5$ (that is, six levels of intimacy, which would map to, for instance, the friend relationship as: 0-no relationship, 1-acquaintance, 2-distant friend, 3-friend, 4-close friend, 5-best friend).

[4]Note that privacy policies defined using any such approach specify the actions that are permitted for particular users and a privacy policy can be reconstructed from the actions permitted for particular users, which is needed to reconstruct privacy policies once an agreement on the actions allowed for particular users is achieved, as we detail later on when the negotiation mechanism is presented. Note also that our mechanism does not even need users to specify their individual privacy preferences for each and every item, users could specify their preferences for groups or categories of items, e.g., users could specify the same preferences for all the photos in a photo album.
[5]Over the course of this article, we shall use relationship strength, tie strength, and intimacy as equivalents.

2.2. Relationship-based Privacy Policies

Privacy policies in relationship-based access control [Carminati et al. 2009; Fong 2011; Wang et al. 2012; Such et al. 2012] consider different relationship types $R = \{r_1, \ldots, r_l\}$ — e.g., family, friends, colleagues, etc. — as well as a mapping $r : U \times U \to R$ so that $r(a, b)$ is the relationship type between users a and b. Privacy policies in relationship-based access control also consider the intimacy (or strength of the relationships) that a user has to other social media users but contextualised within each relationship type. In particular, privacy policies define a different intimacy threshold that users of each relationship type must have with the user that defines the privacy policy to access the specific item[6].

Definition 2.2. A privacy policy is a tuple $P = \langle \theta_1, \ldots, \theta_{|R|}, E \rangle$, where $\theta_j \in [0, \mathcal{Y}]$ is the intimacy threshold for the relationship type $r_j \in R$, and $E \subseteq T$ is the set of exceptions to the policy.

We denote P_a as the preferred privacy policy of the user which agent $a \in Ag$ is representing. For instance, given the set of relationship types $R = \{\texttt{friends}, \texttt{colleagues}, \texttt{family}\}$, negotiating user a could have the privacy policy $P_a = \langle 0, 4, 0, \emptyset \rangle$, so that all friends, close colleagues, and all family members can access the item (assuming $\mathcal{Y} = 5$ as stated above). Another example would be that negotiating user a only wants to share with close friends, but not with colleagues or family. This would be represented with the following privacy policy $P_a = \langle 4, 6, 6, \emptyset \rangle$, so that only close friends can access the item (no one from family and colleagues would have enough intimacy with a as $\mathcal{Y} = 5$ is the maximum possible intimacy in this example). Finally, to showcase the need for exceptions, negotiating user a may have the privacy policy that only close friends can access the item except b, who is also a close friend — e.g., a may be organising a surprise birthday party for b, so that she invites all their close friends to the event but b, as doing otherwise would mean ruining the surprise. This will be mapped to the privacy policy $P_a = \langle 4, 6, 6, \{b\} \rangle$, so that only close friends can access the item (no one from family and colleagues would have intimacy enough) except b, who is also close friend (i.e., assume $int(a, b) \geq 4$) but who is put as an exception to the policy — $P_a.E = \{b\}$.

3. MECHANISM OVERVIEW

The mechanism proposed in this paper takes as inputs the individual privacy policy of each negotiating agent — P_a for all $a \in N$ — and the intimacies among agents — $int(a, b)$ for all $a, b \in Ag$ —, elicited as described in the previous section. The mechanism has two stages, as shown in Figure 1:

(1) The individual privacy policies of the negotiating agents are inspected to identify any conflict, as described in Section 4.
(2) If conflicts are found, then agents run the negotiation mechanism described in sections 5 and 6 to resolve every conflict found.

The output $\vec{o}$ (as described later on) includes all the target agents that are finally granted access to the item after solving all the conflicts found. Note that the output will not be applied straightforwardly. That is, the mechanism is intended to alleviate

[6]Note that even if we represent relationship-based policies in its formal and mathematical form in this paper, with a explicit intimacy threshold, that does not mean that users need to set numerical intimacy threshold values. Indeed, graphical user interfaces for relationship-based access control should be usable [Fogues et al. 2015]. For instance, assuming $\mathcal{Y} = 5$ and the six levels of intimacy stated above, users could just define using a graphical tool they only want to share with *close friends*, so that the tools translates this decision into an intimacy threshold of 4 for the relationship type friends under the hood.

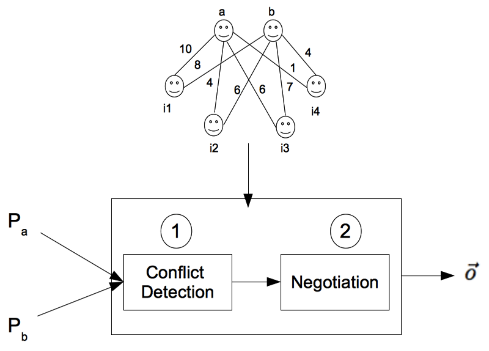

Fig. 1. Mechanism Overview.

user burden to conduct a complex negotiation for each and every possible conflict but not to substitute users decisions and remove control from them. Indeed, it has been widely studied and proved that a fully automated approach to handle users privacy in social media is not desirable in general [Vihavainen et al. 2014], so we also followed the same principle for the particular case of multi-party privacy management. Instead of a fully automated approach, the aim of our proposed mechanism is to automatically find conflicts and suggest a solution. This solution is presented[7] back to the users involved to confirm whether they are happy with it or not. If they are happy with it, then it will be applied. Otherwise, users are free to then enter into a manual negotiation to try to solve the conflict[8].

4. PRIVACY CONFLICT DETECTION

In this section, we describe how conflicts between the preferred privacy policies defined by agents a and b on the item under consideration can be detected. Each agent usually has different intimacies to other agents, so that two privacy policies from two different agents can only be compared in terms of their effects. For instance, suppose that agents a and b have the same preferred privacy policies for a particular item (i.e., these policies have the same intimacy thresholds for the same relationship types). Suppose also that agents a and b have different intimacies to agent i_1 in T. If agent a has an intimacy with i_1 below the corresponding threshold but agent b has an intimacy with i_1 above the threshold, their individual decisions on whether to grant access to i_1 would be different. A similar example can be constructed for the case where the two policies are different but, because of the individual intimacies of agents a and b to other agents in T, they suggest the same decision. Thus, we need to consider the effects that each particular policy has on the set of target agents T to determine whether or not the policies of two negotiating agents are in conflict. That is, we need to know which agents are granted/denied access by a given policy.

Privacy policies dictate a particular action to be performed when an agent in T tries to access the item. Assuming a set of available[9] actions $\Pi = \{0, 1\}$, so that 0 means

[7]A detailed study on how to best present the information to foster trust in the system is out of the scope of this paper and a very interesting line for future research.
[8]Note also that if the inputs of the mechanism might change some time after a solution has been agreed — e.g., a negotiating agent befriends another or a negotiating agent changes its individual privacy policy, the mechanism could run again and another negotiation could take place if new conflicts are created because of the changes.
[9]Mainstream social media (Facebook, Google+, etc.) and relationship-based access control models (like the ones cited above) focus on modelling decisions only about granting/denying access to an item. An interesting

denying access to the item and 1 means granting access to the item, the action to perform according to a given privacy policy is determined as follows:

Definition 4.1. Given an agent $a \in Ag$, its privacy policy $P_a = \langle \theta_1, \ldots, \theta_{|R|}, E \rangle$, an agent $i \in T$, and a set of actions Π, we define the function $act_a : T \to \Pi$, so that[10]:

$$act_a(i) = \begin{cases} 1 & \text{iff } int(a,i) \geq P_a.\theta_{r(a,i)} \wedge i \notin P_a.E \\ 0 & \text{otherwise} \end{cases}$$

We also consider so-called *action vectors* $\vec{v} \in \Pi^n$, i.e. complete assignments of actions to all agents in T, so that $v[i]$ denotes the action for $i \in T$. When a privacy policy is applied to T, it produces such an action vector:

Definition 4.2. The *action vector* induced by privacy policy P_a of negotiating agent a in T is $\vec{v_a}$, where $v_a[i] = act_a(i)$.

We now consider the following problem: agents a and b have their own privacy policies for the same item, and the effect of these policies leads to different action vectors, e.g., there are target agents who are granted access according to one policy but denied access according to the other. In this case, we say that these two privacy policies are *in conflict*:

Definition 4.3. Given agents a and b, their preferred privacy policies for the item under negotiation P_a and P_b, and the action vectors induced by P_a and P_b, which are $\vec{v}_a$ and $\vec{v}_b$ respectively, we say that P_a and P_b *are in conflict* with respect to the item under negotiation iff $\vec{v}_a \neq \vec{v}_b$.

Further, we say that the *agents in conflict* is the set $\mathcal{C} = \{i \in T \mid v_a[i] \neq v_b[i]\}$. The complexity of the conflict detection mechanism is $\mathcal{O}(T \cdot N)$, where T is the number of target agents, and N is the number of negotiating agents both as defined above.

Table I. Intimacies for Example 4.4.

	i_1	i_2	i_3	i_4
a	10	6	4	1
b	8	6	7	4

Example 4.4. Suppose a set of agents $Ag = \{a, b, i_1, i_2, i_3, i_4\}$, a unique relationship type among them $R = \{r_1\}$, and a set of possible actions $\Pi = \{0, 1\}$, with 0/1 meaning denying/granting access to the item. Agents a and b are to decide which agents to grant access to a photo in which both of them are depicted, and the intimacy values of agents a and b toward others are as shown in Table I, with $\mathcal{Y} = 10$. Suppose that agent a would prefer the policy $P_a = \langle 5, \emptyset \rangle$, so that $\vec{v}_a = (1,1,0,0)$ — i.e., agent a wants to grant access to agents i_1 and i_2, toward whom she has an intimacy greater or equal to 5, but not to agents i_3 and i_4 who are less intimate to her. However, agent b would prefer the

path for future research would be to consider other types of access privileges like "*read but not re-share*", etc.

[10]This function will be different depending on the access control model used. For instance, in group-based access control (like Facebook lists) this function could be specified as:

$$act_a(i) = \begin{cases} 1 & \text{if } i \text{ is in a group that is granted access and} \\ & i \text{ is not blocked individually} \\ 1 & \text{if } i \text{ is granted access individually} \\ 0 & \text{otherwise} \end{cases}$$

policy $P_b = \langle 4, \emptyset \rangle$, so that $\vec{v}_b = (1,1,1,1)$ — i.e., agent b wants to grant access to agents i_1, i_2, i_3, and i_4. As $\vec{v}_a \neq \vec{v}_b$, P_a and P_b are in conflict and the set of agents in conflict is $\mathcal{C} = \{i_3, i_4\}$.

5. DEALS AND THEIR UTILITY

When agents run into a conflict, they can still negotiate a common action vector for the item in question to achieve a compromise, even if this will not result in an optimal policy for either of them. Such an outcome (or deal) is simply an action vector $\vec{o} \in \Pi^n$ such that $n =\mid T \mid$, and the negotiation space is the space of all such vectors, which agents can rank according to utility functions that compactly reflect agents' preferences as will be defined in this section. Based on these utility functions, agents will agree on a particular action vector following the negotiation mechanism that we present in Section 6.

5.1. From Deals to Local Privacy Policies

After agents agree on a particular action vector, they must represent it in the form of a local privacy policy, so that users could consult it without having to check who is granted access or not individually. A particular action vector is likely to be represented with different privacy policies for each agent because each of them has different intimacies toward agents in T. Also, it is crucial that the resulting privacy policies are as simple (in terms of the number of exceptions they include) as possible to ensure that users can understand them, which is crucial for an appropriate privacy management [Cranor and Garfinkel 2005]. For instance, a privacy policy that includes 100 exceptions will be far more difficult to read and understand by the user than a privacy policy that only includes one exception. Thus, we are interested in a privacy policy that minimises the number of exceptions among all the privacy policies that can induce the same action vector.

Definition 5.1. Given an action vector $\vec{o}$, the privacy policy that induces $\vec{o}$ in T and minimises the number of exceptions is defined as:

$$P^{\vec{o}} = \arg \min_{P=\langle \theta_1,\ldots,\theta_{|R|},E \rangle,\ \vec{v}=\vec{o}} \mid E \mid$$

Example 5.2. Suppose Example 4.4 and that we would like to obtain the privacy policy for negotiating agent a that induces the action vector b would prefer — i.e., $veco = vecv_b = (1,1,1,1)$ — with the least number of exceptions. The first thing would be to enumerate all privacy policies for negotiating agent a that represent $\vec{o} = (1,1,1,1)$. These would be the following (recall that we are only considering in Example 4.4 one relationship type, hence only one intimacy threshold is required):

— $P_1 = \langle 11, \{i_1, i_2, i_3, i_4\} \rangle$, which means that none of the agents are given access (the maximum intimacy is 10, so adding one means no agent will have enough intimacy with a — this would be equivalent with any number greater than 10), but then i_1 to i_4 are added as exceptions, i.e., they do not have intimacy 11 with a but they will be granted access anyway.

— $P_2 = \langle 10, \{i_2, i_3, i_4\} \rangle$, which means that only agents with at least intimacy 10 are given access, but then i_2 to i_4 are added as exceptions, i.e., they do not have intimacy 10 with a but they will be granted access anyway.

— $P_3 = \langle 6, \{i_3, i_4\} \rangle$, which means that only agents with at least intimacy 6 are given access — i.e., i_1 and i_2, but that i_3 and i_4 are exceptions to this, so they are granted access as well.

— $P_4 = \langle 4, \{i_4\} \rangle$, which means that only agents with at least intimacy 4 are given access — i.e., i_1, i_2, and i_3; but that i_4 is an exception to this.

— $P_5 = \langle 1, \emptyset \rangle$, which means that only agents with at least intimacy 1 are given access — i.e., i_1, i_2, i_3, and i_4; with no exceptions.

All of the above individual privacy policies for negotiating agent a would be equivalent, as all of them would induce exactly the same action vector $(1, 1, 1, 1)$. However, P_5 has no exceptions, so it is the one that would be chosen as $P_a^{\vec{o}}$ — i.e., the individual privacy policy of a that induces $\vec{o}$ with the least number of exceptions, to facilitate a's comprehension and understandability of resulting privacy policies [Cranor and Garfinkel 2005].

5.2. Utility Function

Intuitively, the rationale of the utility function agents will use to rank possible negotiation outcomes is:

(1) An outcome will be ranked based on the distance (in terms of intimacy) between the agent's preferred privacy policy and the privacy policy that induces the outcome, establishing an intimacy-based ordering of the outcomes as suggested by [Wiese et al. 2011]. That is, the farther the privacy policy that induces the outcome is from the agent's preferred privacy policy, the less valued the outcome will be.
(2) An outcome will be ranked according to the number of exceptions of the privacy policy that induces the outcome, so that privacy policies should include as few exceptions as possible to ensure readability and understandability [Cranor and Garfinkel 2005]. That is, the more exceptions the privacy policy that induces the outcome entails, the less valued the outcome will be.

We start defining the intimacy distance between two policies. Privacy policies may have different intimacy dimensions (i.e., one per each relationship type considered), so that a metric in the $\mathbb{R}^{|R|}$ space is needed to compare them. We use the Euclidean distance to measure the distance between two policies as follows:

Definition 5.3. Given two policies P and Q the distance between them is[11]:

$$d(P, Q) = \sqrt{\sum_{r \in R} (P.\theta_r - Q.\theta_r)^2}$$

The advantage of using Euclidean distance is that it is sensitive to large variations in one dimension (relationship type). For instance, for two policies with a large difference in the "friends" relationship type (e.g., one of them grants any acquaintances access and the other only grants access to close friends) the distance would be large as well. This aligns with empirical evidence that suggests that intimacy distance plays a significant role on deciding to disclose or not [Green et al. 2006; Strahilevitz 2005; Houghton and Joinson 2010; Wiese et al. 2011], so the higher the intimacy distance between what the user would like for a particular relationship type and the intimacy between that user and a target user, the less the user would like to share with the target use[12].

[11]An example of this function for group-based access control models could be the euclidean distance but considering, for each possible group (instead of for each possible relationship type), the distance between the minimum intimacy of the users in this group that are granted access in the two policies compared (instead of the difference between intimacy thresholds).
[12]Other metrics that can be used to compare vectors in the same space, such as the average of differences for each relationship type, the Chebyshev distance or the Manhattan distance do not always align with [Green et al. 2006; Strahilevitz 2005; Houghton and Joinson 2010; Wiese et al. 2011]. A simple average of the differences of intimacy in each relationship type, would lead to the final distance value being significantly attenuated if the differences in other relationship types (e.g., "family","colleagues", etc) are low. The

Example 5.4. Suppose Example 4.4 again and that, as stated for from Example 5.2 above, for negotiating agent a, the privacy policy that induces $\vec{o} = \{1,1,1,1\}$ with the least number of exceptions is $P_a^{\vec{o}} = \langle 1, \emptyset \rangle$. Therefore, the distance between negotiating agent a's preferred privacy policy P_a and $P_a^{\vec{o}}$ is the following — recall we were only considering in Example 4.4 one relationship type (r_1):

$$d(P_a, P_a^{\vec{o}}) = \sqrt{\sum_{r \in R=\{r_1\}} (P_a.\theta_r - P_a^{\vec{o}}.\theta_r)^2} = \sqrt{(P_a.\theta_{r_1} - P_a^{\vec{o}}.\theta_{r_1})^2} = \sqrt{(6-1)^2} = 5$$

We now define the utility function based on (i) the privacy policy that minimises the number of exceptions among all the privacy policies that induce the same action vector (Def. 5.1) according to [Cranor and Garfinkel 2005] and (ii) the intimacy distance (Def. 5.3) according to [Green et al. 2006; Strahilevitz 2005; Houghton and Joinson 2010; Wiese et al. 2011]; so that the higher the number of exceptions/the intimacy distance, the less utility.

Definition 5.5. Given agent a and its preferred privacy policy P_a, the utility of an action vector $\vec{o}$ for agent a is:

$$u_a(\vec{o}) = \lambda_a(P^{\vec{o}}) \cdot (\mathcal{D} - d(P_a, P^{\vec{o}}))$$

In this equation, $\mathcal{D}$ accounts for the maximum possible distance between two privacy policies, which would be obtained if the difference was $\mathcal{Y}$ for all the relationship types, and

$$\lambda_a(P^{\vec{o}}) = 1 - \frac{\mid P^{\vec{o}}.E \mid}{\mid T \mid}$$

accounts for the number of exceptions that $P^{\vec{o}}$ — the privacy policy that induces $\vec{o}$ in T with the minimum possible exceptions, as defined above — would entail (denoted as $\mid P^{\vec{o}}.E \mid$) with respect to the maximum number of exceptions possible, i.e., the number of target agents $\mid T \mid$.

The main factor of the utility function is the distance between the policies, which tells us how happy the user will be with the final result. If a potential solution is very far from what the user would prefer, it does not matter whether the solution has more or less exceptions, the solution will not be acceptable as it will be too far from the user individual privacy preferences (expressed as her individual privacy policy) according to existing evidence [Wiese et al. 2011]. Although not the main factor, there is also another minor factor to be considered, which is related to how easy would be for users

Chebyshev distance is the maximum of all distances in each dimension. This would work well for the case in which there is a large variation on only one dimension (which would be the maximum), but if there are large variations in more than one dimension, this would not be accounted as only the maximum would be considered. In contrast, the Euclidean distance would clearly signal this difference, so that the final distance would be higher for the case in which there are large variations in more than one dimension. Finally, the Manhattan distance is the sum of the differences for all dimensions. Clearly, we could have small to medium variations in some dimensions that would add up to the same value as if only one dimension had a large variation. For instance, small variations in relationship types friends, family, colleagues, etc, could give the same result that a large variation in only one dimension (grant access to all your work colleagues instead of only to those you are closer to you), which would be clearly worse in terms of privacy implications. Therefore, the intimacy distance would be unable to catch these nuances, which might impact on how the utility function is able to model user preferences in this domain.

to understand the proposed solution as a privacy policy, so users can actually take the decision of whether they are happy with the solution, minimising potential errors and misunderstandings [Cranor and Garfinkel 2005]. This is why the main distance factor is corrected with another factor λ_a with values within the [0,1] interval according to the ratio of exceptions. Note, however, that this correction will be negligible most of the time, because: i) from all the equivalent privacy policies that induce exactly the same action vector $\vec{o}$ — i.e., they all induce the same negotiation outcome, we are only considering the one that minimises the number of exceptions according to Definition 5.2; and ii) the average user has around 130 friends [Quercia et al. 2012], so the number of target users can be quite big, but only when the number of exceptions is also very high, then the ratio may have an impact on the distance.

Example 5.6. Suppose again Example 4.4, and that we would like to know the utility for negotiating agent a for action vector $\vec{o} = \{1, 1, 1, 1\}$. From Example 5.2, we know that $P_a^{\vec{o}} = \langle 1, \emptyset \rangle$ and from Example 5.4, we know that $d(P_a, P_a^{\vec{o}}) = 5$. As $P_a^{\vec{o}}$ does not have any exception, then $\lambda_a(P_a^{\vec{o}}) = 1 - \frac{|P_a^{\vec{o}}.E|}{|T|} = 1 - \frac{0}{4} = 1$. Moreover, from Example 4.4, $Y = 10$ and there is only one relationship type $R = \{r_1\}$, so the maximum possible intimacy distance is $D = 10$. Therefore, the utility for negotiating agent a for action vector $\vec{o} = \{1, 1, 1, 1\}$ is:

$$u_a(\vec{o}) = \lambda_a(P_a^{\vec{o}}) \cdot (\mathcal{D} - d(P_a, P_a^{\vec{o}})) = 1 \cdot (10 - 5) = 5$$

6. NEGOTIATION PROTOCOL

Next, we consider how a mutually acceptable action vector $\vec{o}$ can be agreed upon by two negotiating agents $a, b \in N$ for the particular item under negotiation. In order for a and b to be able to negotiate a common action vector for a given item, we need to define a negotiation mechanism. A negotiation mechanism is composed of: (i) a negotiation protocol, which is a means of standardising the communication between participants in the negotiation process by defining how the actors can interact with each other; and (ii) strategies that agents can play over the course of a negotiation protocol [Rosenschein and Zlotkin 1994]. Negotiation protocols determine the strategies agents can play during the execution of the negotiation protocol. Although there are some negotiation protocols proposed in the related literature [Lopes et al. 2008], not all of them comply with the requirements for the domain we are tackling in terms of the strategies they permit. In particular, the requirements are:

— The protocol must permit negotiation strategies that are stable. A stable strategy is one whereby if one agent is playing it the others' best strategy is to also play the same strategy. This is very important, as it is expected that each agent usually cares only about her user's own utility and will always try to play the strategy that can get the highest utility, even if it means everyone else is much worse [Vidal 2010]. This could clearly lead to outcomes in which one agent gets an extremely high utility and everyone else gets almost nothing, which would be unfair. Therefore, stability is a very desirable property for our domain, as the negotiation mechanism should be fair, so that one user cannot just impose her preferences on the others.

— The protocol must permit negotiation strategies that converge to the optimal solution (what is know in the negotiation literature as *efficiency*). This is of crucial importance because of two main reasons: (i) if the negotiation protocol does not allow negotiation strategies that make the negotiation converge, agents could keep negotiating forever; (ii) the preferences of all the negotiating agents' users should be respected as much as possible.

The simplest negotiation protocol that has these properties is the *one-step protocol* [Rosenschein and Zlotkin 1994]. This protocol has only one round, where the two agents propose a deal, and they must accept the deal that maximises the product of both agents' utilities. In case deals have the same product, one of them is chosen randomly (e.g. by flipping a coin). In our case, a deal is an action vector. Thus, each agent will propose an action vector and they will accept the one that maximises the product of their utilities. To calculate individual utilities, agents use the utility function presented in Section 4, which, in turn, uses the agents' preferred privacy policies elicited to detect conflicts (as explained in Section 3), and agents' intimacies, which can be accurately estimated from content already published (as explained in Section 2).

It was formally proven that the best strategy that agents can follow in this protocol is to propose the deal (action vector) that is best for themselves amongst those with maximal product of utilities. This strategy is both stable and efficient [Rosenschein and Zlotkin 1994]. Its stability derives from being a Nash equilibrium (if one of the two agents follows this strategy the other agent's best strategy is to also follow this strategy), no agent has anything to gain by changing only her own strategy unilaterally. It is efficient in the sense that if there exists a solution the agents will find it using this strategy. Other negotiation protocols such as the well-known monotonic concession protocol [Rosenschein and Zlotkin 1994] or the alternating offers protocol [Osborne and Rubinstein 1990] do not allow strategies for this domain that would be both stable and efficient. The monotonic concession protocol is known to only allow strategies that can be either stable or efficient but not both [Endriss 2006]. The alternating offers protocol has no convergence guarantees in its basic form, and for the time-dependent form utilities must be time-dependent [Vidal 2010], which is not the case in this domain.

Table II. Action vector utilities for Example 6.1. Note the preferred individual privacy policy of each negotiating agent is always the same and as described in Example 4.4, i.e., $P_a = \langle 6, \emptyset \rangle$ and $P_b = \langle 4, \emptyset \rangle$.

$\vec{o}$	$P_a^{\vec{o}}$	$P_b^{\vec{o}}$	λ_a	λ_b	$d(P_a, P_a^{\vec{o}})$	$d(P_b, P_b^{\vec{o}})$	$u_a(\vec{o})$	$u_b(\vec{o})$	$u_a(\vec{o}) \times u_b(\vec{o})$
(1,1,0,0)	$\langle 6, \emptyset \rangle$	$\langle 6, \{i_3\} \rangle$	1	0.75	0	2	10	6	60
(1,1,0,1)	$\langle 1, \{i_3\} \rangle$	$\langle 4, \{i_3\} \rangle$	0.75	0.75	5	0	3.75	7.5	28.125
(1,1,1,0)	$\langle 4, \emptyset \rangle$	$\langle 6, \emptyset \rangle$	1	1	2	2	8	8	**64**
(1,1,1,1)	$\langle 1, \emptyset \rangle$	$\langle 4, \emptyset \rangle$	1	1	5	0	5	10	50

Example 6.1. Consider the application of our negotiation mechanism to resolve the conflicts detected in Example 4.4. As the set of agents in conflict was $\mathcal{C} = \{i_3, i_4\}$, the possible action vectors that could be selected as a compromise are the ones in Table II under the $\vec{o}$ column. Agents a and b will be able to rank each of the possible action vectors according to their preferences by measuring the utility of the action vectors, using the utility function from Definition 5.5, which is calculated based on the preferred privacy policies of each agent and the intimacies between the negotiating agents and other target agents (shown in Table I). Table II shows, for each action vector, the privacy policy with the least number of exceptions that induces this action vector for each negotiating agent, the ratio of exceptions for each negotiating agent, the distance between the preferred individual privacy policy of each negotiating agent to the privacy policy that induces the action vector, the utilities for each individual agent as well as the product of both agents' utilities. For instance, the utility of $\vec{o} = (1, 1, 1, 0)$ for a is $u_a(\vec{o}) = \lambda_a(P_a^{\vec{o}}) \cdot (\mathcal{D} - d(P_a, P_a^{\vec{o}}))$, where:

— $\mathcal{D} = 10$ because we only have one relationship type in this example, so the maximum distance possible will be the maximum possible intimacy value $\mathcal{Y} = 10$ in our example.

— $P_a^{\vec{o}} = \langle 4, \emptyset \rangle$ because this is the privacy policy that represents the action vector $\vec{o} = (1,1,1,0)$ for agent a with the minimum number of exceptions. That is, all target agents with an intimacy higher or equal than 4 are granted access to the item under consideration.
— $\lambda_a(P_a^{\vec{o}}) = 1$, because $P_a^{\vec{o}}$ does not entail any exception.
— $d(P_a, P_a^{\vec{o}}) = \sqrt{(P_a.\theta - P_a^{\vec{o}}.\theta)^2} = \sqrt{(6-4)^2} = 2$.

Thus, $u_a(\vec{o} = (1,1,1,0)) = 1 \times (10-2) = 8$. Finally, agents will propose the action vector that is most favourable to them in terms of maximising the utility product. In our example, both will propose $\vec{o} = (1,1,1,0)$, which will be the outcome of the negotiation. This is because the best deal in this case is the one that makes both negotiating agents lose the same utility to achieve a compromise, i.e., negotiating agent a makes its individual privacy policy a bit more open and negotiating user b makes its individual privacy policy a bit more closed. The benefits are that negotiating user b is still able to share the item with i_3 (which seems to be a very good friend of b — i.e., $int(b, i_3) = 7$), while she accepts not to share with i_4 (which seems to be just a regular friend of b — i.e., $int(b, i_4) = 4$); and the benefits for negotiating agent a is that she avoids sharing with i_3 (who is clearly a distant acquaintance — i.e., $int(a, i_4) = 1$ — and way too far from what a would like as $P_a^{\vec{o}}.\theta = 6$), and only shares with i_3 (who is very close to the intimacy threshold a has in its individual privacy policy — i.e., $int(b, i_3) = 4$).

6.1. Complexity

The number of possible deals in this setting is exponential in the number of agents in $\mathcal{C}$, the set of agents in conflict. This is because the negotiation mechanism will have to consider a number of actions vectors equal to k^l permutations in order to find the one that maximises the product of utilities, where k is the number of possible actions in Π and l is the number of the number of agents in conflict $\mathcal{C}$. The number of agents in $\mathcal{C}$ will change from negotiation to negotiation — it completely depends on the preferred policies of the negotiating agents a and b, so that we cannot predict the number of action vectors to be considered for each particular case *a priori*, even though it seems clear that the more target agents in T the more possibilities for the number of agents in $\mathcal{C}$ to be higher. In the worst case, all of the target agents T will be in $\mathcal{C}$, i.e. the problem complexity grows exponentially in the number of target agents. Moreover, each possible outcome needs to be evaluated from the point of view of the other negotiating agents. Therefore, the complexity of the negotiation mechanism would be $\mathcal{O}(k^l \cdot n)$, where $k = \mid \Pi \mid$ is the number of possible actions, $l = \mid \mathcal{C} \mid$ is the number of conflicts, and $n = \mid N \mid$ is the number of negotiating agents. Finally, the worst case would be an upper bound for the complexity, so that in the worst case the complexity would be $\mathcal{O}(k^m \cdot n)$, where $m = \mid T \mid$ is the number of target agents.

7. HEURISTICS

To tackle the exponential blowup in the number of possible deals, we firstly considered *complete* approaches that would decrease the complexity while always finding the *optimal* solution. In particular, we considered approaches such as dynamic programming (and other divide and conquer algorithms) or branch and bound algorithms. However, none of these approaches was suitable for this domain. This is because the problem we are tackling does not exhibit the overlapping subproblems and optimal substructure properties required for a dynamic programming approach, and we were not able to develop a complete BnB algorithm because we were unable to find good-enough upper bounds in this domain for the utility of partial action vectors. For instance, using the utility function defined in Section 5 while ignoring target agents for whom no decision has been made (the one used for the Greedy heuristic) turned out to be too optimistic to prune enough nodes from the search space, so that the resulting BnB algorithm was not efficient enough to be used in practice.

As *complete* approaches were not possible, we decided to develop heuristics — i.e., *incomplete* approaches — that could reduce the number of action vectors considered when maximising the utility product to only those that appear most promising. This obviously involves a risk of losing optimality — we empirically prove later on in Section 8 that these optimality losses remain within acceptable levels in practice.

7.1. Distance-based Heuristic

Our first heuristic fixes the action to be taken for some conflicting target agents without trying both possible actions (granting or denying access) when generating the possible action vectors.

Algorithm 1 Distance-based Heuristic - Agent a

```
Input: $T$, $\vec{v}$, $\vec{w}$, $P_a$, $P_b$, $\varphi$
Output: $\vec{o}$
 1: $maxval \leftarrow 0$
 2: $initialize\_to\_zeros(\vec{o})$
 3: $initialize\_to\_zeros(\vec{t})$
 4: for all $i \in T$ do                              ▷ Generating a partial action vector
 5:     if $v_a[i] = v_b[i]$ then
 6:         $t[i] \leftarrow v[i]$
 7:     else                                          ▷ Conflict - applying heuristic
 8:         $d_a \leftarrow\mid P_a.\theta_{r(a,i)} - int(a,i) \mid$
 9:         $d_b \leftarrow\mid P_b.\theta_{r(b,i)} - int(b,i) \mid$
10:         if $\mid d_a - d_b \mid\geq \varphi$ then
11:             if $d_a > d_b$ then
12:                 $t[i] \leftarrow v_a[i]$
13:             else
14:                 $t[i] \leftarrow v_b[i]$
15:             end if
16:         else
17:             $t[i] \leftarrow *$                   ▷ Both will be considered
18:         end if
19:     end if
20: end for
21: for all $\vec{x} \in \mathcal{X}^{\vec{t}}$ do     ▷ Utility product maximisation
22:     $prod \leftarrow u_a(\vec{x}) \times u_b(\vec{x})$
23:     if $prod > max$ then
24:         $\vec{o} \leftarrow \vec{x}$
25:         $max \leftarrow prod$
26:         $maxUT \leftarrow u_a(\vec{x})$
27:     end if
28:     if $prod = max$ and $u_a(\vec{x}) > maxUT$ then
29:         $\vec{o} \leftarrow \vec{x}$
30:         $maxUT \leftarrow u_a(\vec{x})$
31:     end if
32: end for
```

Informally speaking, the heuristic calculates how *important* assigning a particular action to a particular target agent is for one of the negotiating agents. This importance is calculated by measuring the intimacy distance between a target agent and the threshold for the relationship type of this agent in the preferred privacy policies of both agents a and b. If the difference between these two distances is higher than a threshold value φ — the so-called *importance threshold* — we consider that the action for that target agent is more important to the negotiating agent with higher distance than to the negotiating agent with the lower distance. Thus, we only consider action vectors in which the particular target agent is assigned the action suggested by the negotiating agent with the highest distance. Both negotiating agents are using the same φ value.

Algorithm 1 illustrates the proposal generation process with the heuristic for agent a (the pseudocode for agent b would be the same except for lines 27-30 as explained below). In this algorithm, we use a so-called *partial action vector* $\vec{t}$ with $t_i \in \{0, 1, *\}$, where 0/1 means that action vectors generated from that partial action vector will deny/grant target agent i access and $*$ means that both actions will be considered when generating action vectors. For each target agent i, if both action vectors $\vec{v}$ and $\vec{w}$ assign the same action — i.e. there is no conflict in the action to be taken for this particular agent, then this action is chosen for the partial action vector. If the action vectors do not assign the same action, we have a conflict. In case of conflict, we first measure the distances in intimacy of agent i to both agents a and b (Lines 11 and 12). If the difference between these two distances is greater than or equal to the *importance threshold* φ, we apply the action corresponding to the greater distance. If not, we assign a $*$ value to consider both alternatives. In section 8, we discuss the effect of different *importance thresholds* φ on the performance of the negotiation mechanism.

After the creation of the partial action vector, we only consider action vectors that comply with the partial action vector in the utility product maximisation step (from Line 21). This set is defined as $\mathcal{X}^{\vec{t}} = \{\vec{x} \mid \forall i \in T,\ x_i = t_i \vee t_i = *\}$. That is, we are constraining the set of action vectors considered when maximising the product of utilities for both agents. In Lines 22-26, if the product of utilities for the current action vector is higher than for the best action vector seen so far, agent a updates the latter with the current one. In Lines 27-30, if the product of utilities for the current action vector is equal to the product of utilities of the best action vector seen so far and the individual utility for agent a is higher for the current action vector than for the best action vector seen so far, agent a updates the best action action vector with the current one. This is because, as explained in the previous section, the best strategy that agents can follow in this protocol is to propose the deal (action vector) that is best for themselves among those with a maximal product of utilities. Lines 26, 28, and 30 will change for agent b to consider its individual utility instead of the individual utility of agent a.

Example 7.1. Suppose Example 4.4 and assume an importance threshold of $\varphi = 2$. We start constructing the partial action $\vec{t}$ with the target agents that were not in conflict, i.e., $t[i_1] = 1$ and $t[i_2] = 1$. After this, we consider i_3. This is a conflict, so we calculate $d_a = 6 - 4 = 2$ and $d_b = 7 - 4 = 3$. Then, $\mid d_a - d_b \mid = 1$, but the importance threshold is greater than this ($\varphi = 2$). Therefore, we do not take any solution at this point and set $t[i_3] = *$. For the other conflicting target agent i_4, we have that $d_a = 6-1 = 5$, $d_b = 4-4 = 0$, and $\mid d_a - d_b \mid = 5$, which in this case is more than the importance threshold ($\varphi = 2$), so we solve the conflict by setting $t[i_4] = 0$ (which is the action suggested by a, as $d_a \geq d_b$). The resulting partial action vector is $\vec{t} = (1, 1, *, 0)$, so $\mathcal{X}^{\vec{t}} = \{(1, 1, 0, 0), (1, 1, 1, 0)\}$ — i.e., we now perform a *complete* enumeration of the possible action vectors resulting from the partial action vector. According to Table II, we know that $u_a(1, 1, 1, 0) \times u_b(1, 1, 1, 0)$ is greater than $u_a(1, 1, 0, 0) \times u_b(1, 1, 0, 0)$, therefore we will get into the same optimal solution as with the mechanism without heuristic $\vec{o} = (1, 1, 1, 0)$. Note that for this example, the distance-based heuristic will always produce the optimal solution, regardless of the importance threshold φ used, because when setting $t[i_3]$, $\mid d_a - d_b \mid = 1$. We have already seen the case when $\varphi > 1$, and for when $\varphi \leq 1$, then $t[i_3] = 1$ because then $d_b > d_a$, so the action suggested by b would be taken, so the resulting action vector will always be $\vec{o} = (1, 1, 1, 0)$, which is the optimal solution.

7.2. Greedy Heuristic

Our second heuristic follows a greedy approach to propose an action (granting/denying access) for each conflict separately. The informal idea is to make the locally optimal choice at each stage with the hope of finding a global optimum.

Like the previous heuristic, this one also considers partial action vectors. Here, however, agents also calculate the utility of partial action vectors, because the greedy heuristic needs an estimation of how good the partial action vectors generated in each step of the heuristic are. To this aim, the utility function defined in Section 5 is used while ignoring target agents for whom no decision has been made so far, i.e. where the partial action vector contains $*$ at the entry corresponding to the respective target agent. This is done by considering that the action assigned to them is the most desired by each negotiating agent. In this way, the utility of a partial action vector acts as an upper bound of the utility that will be achieved when having a complete action vector.

Algorithm 2 Greedy Heuristic - Agent a

```
Input: $T$, $\vec{v}$, $\vec{w}$, $P_a$, $P_b$
Output: $\vec{o}$
 1: for all $i \in T$ do                                   ▷ Detecting Conflicts
 2:     if $v[i] = w[i]$ then
 3:         $o[i] \leftarrow v[i]$
 4:     else
 5:         $o[i] \leftarrow *$
 6:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$
 7:     end if
 8: end for
 9:
10: while $\mathcal{C} \neq \emptyset$ do
11:     $maxVal \leftarrow 0$
12:     $\vec{x} \leftarrow \vec{o}$
13:     for all $i \in \mathcal{C}$ do                    ▷ For all the remaining conflicts
14:         for $action \in \Pi$ do
15:             $x[i] \leftarrow action$
16:             $prod = u_a(\vec{x}) \times u_b(\vec{x})$
17:             if $prod > max$ then
18:                 $max \leftarrow prod$
19:                 $maxUT \leftarrow u_a(\vec{x})$
20:                 $maxTarget \leftarrow i$
21:                 $maxAction \leftarrow x[i]$
22:             end if
23:             if $prod = max$ and $u_a(\vec{x}) > maxUT$ then
24:                 $maxUT \leftarrow u_a(\vec{x})$
25:                 $maxTarget \leftarrow i$
26:                 $maxAction \leftarrow x[i]$
27:             end if
28:         end for
29:         $x[i] \leftarrow *$
30:     end for
31:     $o[maxTarget] = maxAction$
32:     $\mathcal{C} \leftarrow \mathcal{C} \setminus \{maxTarget\}$
33: end while
```

The heuristic starts by considering the action vector in which all agents not in conflict are assigned the corresponding (commonly agreeable) action, and all other agents are assigned a $*$ value. Thus, this first partial action vector always has the maximum product of utilities possible with respect to the other possible partial action vectors and complete action vectors. This is because if we ignore all conflicts, the actions assigned to the target agents will completely comply with the preferred privacy policies of the negotiating agents.

After this, the heuristic incrementally assigns an action to one conflict at a time. The choice of the conflict and the action taken to resolve it is made as follows: the heuristic compares all possible grant/deny configurations over all conflicts and greedily chooses the most promising option, i.e. the one that decreases the total product of utilities by the smallest amount. This process is repeated until all conflicts are resolved and a complete action vector has been produced.

Algorithm 2 shows the pseudocode for the greedy heuristic for agent a (the pseudocode for agent b would be the same except lines 19, 23, and 24 would change for agent b to consider its individual utility instead of the individual utility of agent a). We first detect conflicts and create a partial action vector by assigning either the corresponding action to the agents not in conflict or $*$ to the agents that are in conflict (lines 1-8). Then, while there are conflicts that have not been dealt with (line 10), we assign an action to a conflict, one conflict at a time. To achieve this, we try all possible partial action vectors by exploring 0 and 1 values for each of the conflicts (lines 13-24). Then, we select the conflict and action that maximise the product of utilities. Finally, we update the partial action vector with the selected action for the conflict in question and mark that conflict as resolved (lines 31-32).

Example 7.2. Suppose Example 4.4. After initialisation, we have that $\vec{o} = (1,1,*,*)$, as i_3 and i_4 are in conflict. Then, for each conflict, two partial action vectors are constructed with the two possible actions ($\Pi = \{0,1\}$) obviating any other conflict. That is, for i_3 partial action vectors $(1,1,0,*)$ and $(1,1,1,*)$ are generated, and for i_4 partial action vectors $(1,1,*,0)$ and $(1,1,*,1)$ are generated. Then, we pick the most promising partial action vector (the one with the highest product of utilities) $(1,1,1,*)$. To calculate the product of utilities of a partial action vector such as $(1,1,1,*)$, we remove all the target agents with $*$ —in this case we remove i_4 — and calculate the product of utilities for the resulting full action vector as stated above — in this case $u_a(1,1,1) \times u_b(1,1,1)$. Then, the process is repeated and action vectors $(1,1,1,0)$ and $(1,1,1,1)$ are generated from the most promising partial action vector in the previous round $(1,1,1,*)$. As the product of utilities is higher for $(1,1,1,0)$ than for $(1,1,1,1)$, then $\vec{o} = (1,1,1,0)$ is chosen with product of utilities 64 shown Table II, which is the optimal solution that would be obtained without applying the heuristic as well.

7.3. GreedyBnB Heuristic

Our third heuristic is GreedyBnB and it is loosely based on Branch and Bound (BnB) algorithms. These algorithms systematically discard large subsets of fruitless candidates from the search space by means of upper and lower estimated bounds of the quantity being optimised. Our GreedyBnB heuristic operates in a similar way to a BnB algorithm[13]. Informally speaking, the GreedyBnB heuristic explores other branches of the search space different from the one followed by the greedy heuristic. The aim is to explore branches of the search space that were discarded by the greedy heuristic at early stages and that could lead to better outcomes. The GreedyBnB heuristic uses the greedy heuristic as a selection algorithm to prioritise options. In particular, our GreedyBnB uses the Greedy heuristic to estimate a lower bound for the utility of par-

[13]As explained above, a complete BnB algorithm was not implemented because we were unable to find good-enough upper bounds in this domain for the utility of partial action vectors. Nonetheless, we followed some of the BnB principles to develop the GreedyBnB heuristic, but of course, as a heuristic this approach is incomplete — there are no guarantees that the *optimal* solution will always be found.

tial action vectors. That is, given a partial action vector, we estimate its utility to be, at least, the utility of the solution obtained when the greedy heuristic is applied to that partial action vector.

Algorithm 3 GreedyBnB Heuristic - Agent a

```
Input: $T$, $\vec{v}$, $\vec{w}$, $P_a$, $P_b$
Output: $\vec{o}$
 1: for all $i \in T$ do                                         ▷ Detecting Conflicts
 2:     if $v[i] = w[i]$ then
 3:         $t[i] \leftarrow v[i]$
 4:     else
 5:         $t[i] \leftarrow *$
 6:     end if
 7: end for
 8:
 9: $\{\vec{o}, maxVal\} = greedySolution(\vec{t})$              ▷ Obtain a solution with greedy heuristic
10: $L.add(\{\vec{t}, \vec{o}, maxVal\})$
11:
12: while $L$ is not empty do                                    ▷ Explore other possible solutions
13:     $\{\vec{x}, \vec{y}, ut\} \leftarrow L.removeFirst()$
14:     if $ut > maxVal$ or $(ut = maxVal$ and $u_a(\vec{y}) > u_a(\vec{o}))$ then
15:         $maxVal \leftarrow ut$
16:         $\vec{o} \leftarrow \vec{y}$
17:         $L.prune(maxVal)$
18:     end if
19:     for all $i \in$ conflicts($\vec{x}$) do
20:         for $action \in \Pi$ do
21:             $x[i] \leftarrow action$
22:             $\{ut, \vec{y}\} = greedySolution(\vec{x})$
23:             if $ut > maxVal$ or $(ut = maxVal$ and $u_a(\vec{y}) > u_a(\vec{o}))$ then
24:                 $L.add(\{\vec{x}, \vec{y}, ut\})$
25:             end if
26:         end for
27:         $x[i] \leftarrow *$
28:     end for
29: end while
```

Algorithm 3 lists our GreedyBnB heuristic. Firstly, it constructs a partial action vector $\vec{t}$ with all the conflicts detected (Lines 1-7). Secondly, it uses the greedy heuristic to obtain a solution to the partial action vector created (Line 9). Then, it adds the partial action vector, the greedy solution and its utility to the list L (Line 10), which is ordered by decreasing values of utility of the greedy solution obtained. While L is not empty, we retrieve the first element of the ordered list. If this has higher utility than the best solution seen so far, the solution is recorded as the best one seen so far and all the nodes in the list that have less utility are pruned (Lines 14-18). After this, we generate all the possible partial action vectors that arise from considering the possible actions (granting/1 or denying/0) for all the remaining conflicts in the partial action vector retrieved from the list (Lines 19-28). If the greedy solution for any of these partial action vectors produces a higher utility than the best solution seen so far, we add this partial action vector (which will have one conflict less than the partial action vector retrieved from the list), its greedy solution and the utility of the solution to the ordered list (Lines 22-25).

Example 7.3. Suppose Example 4.4. The first step is to create partial action vector $\vec{t} = (1, 1, *, *)$, resulting from i_3 and i_4 being conflicts. After this, we get a solution using the Greedy heuristic, which would be $\vec{o} = (1, 1, 1, 0)$ with 64 as the product of utilities as discussed in Example 7.2. Then, we add $\{(1, 1, *, *), (1, 1, 1, 0), 64\}$ to the list L. While

L is not empty, we extract its first element and update the best solution seen so far if need be. In this case, the first element will be the one we just added so it is also the best solution seen so far. Then, for each of the conflicts not solved yet (i_3 and i_4), all possible partial action vectors are generated — i.e., $(1,1,0,*)$, $(1,1,1,*)$, $(1,1,*,0)$, $(1,1,*,1)$ — and a greedy solution is obtained for each partial action vector. If the greedy solution of any of these partial action vectors is better than the best solution seen so far, then we add them to the list L. In this example, none of the partial action vectors is better than the greedy solution, which we already know is the optimal, so no new elements are added into the list L, and the algorithm will return $\vec{o} = (1,1,1,0)$, which is the same (optimal) solution that would have been obtained had the mechanism without heuristics been applied.

Finally, it is worth noting that the GreedyBnB heuristic can also be implemented in a similar way to anytime algorithms. These algorithms can be interrupted before they end but will still provide a valid solution if interrupted, and are expected to produce increasingly good solutions the more time they are given to run. In particular, we could implement the heuristic so that it stops after a given amount of time, and this is explored further below in Section 8. When the heuristic stops, it will return the best solution seen so far. In the experiments section we describe the performance achieved with the heuristic at various cutoff points.

8. EXPERIMENTAL RESULTS

We conducted a series of experiments to compare the performance of our proposed negotiation mechanism with and without heuristics empirically by measuring the number of action vectors considered and the execution time required to find a solution, as well as the maximum product of utilities obtained.

8.1. Experimental Setting

We implemented our mechanism and heuristics in Java and report experiments conducted on a 3.1 GHz Intel Core i5 iMac with 8 GB RAM. The rationale for the parameters explained below was the following: wherever there was a real value or distribution of one of the parameters in real Social Media, we used that to inform the values considered; and if there was no information about the real value of distribution of the parameters considered, then we used a different random value for each experiment conducted. In all the experiments, we performed 1000 different simulations in order to support the findings with statistically significant evidence.

We considered 3 relationship types, which is in line with related literature on online communities and social networks [Raad et al. 2013; Hogg et al. 2008]. Besides, we considered different numbers of target agents — 10 to 200 in increments of 10 — based on typical real-world values, given that the average user on Facebook has 130 friends [Quercia et al. 2012]. We also considered a maximum intimacy value $\mathcal{Y}$ of 10, i.e. an intimacy range of [0,10]. Note that the maximum intimacy value does not have any effect on the performance of the mechanism, as it just defines a range of real values for the possible intimacy values. If another range of real values is used, the intimacy values would be scaled but the results would be the same in terms of the solution agreed.

For the parameters for which there is no real data or information about their distribution, we chose to randomise them to minimise the introduction of any bias and to allow us to repeat the experiments multiple times to understand the behaviour of the mechanism under different conditions. That is, we generated a random matrix of intimacies among agents, random assignment of agents to the 3 relationship types considered, and random privacy policies for both negotiating agents. Moreover, we ensured that for each situation, there was at least one conflict — if not, we reinitialised the matrix and privacy policies until at least one conflict occurred.

After this, we ran the negotiation mechanism with and without heuristics to obtain the solution to this situation and accounted for the number of action vectors each approach needed to explore and, where available, the utility of the solution achieved and the execution time needed to achieve the solution. As explained below, there were situations in which not all the approaches were able to produce an actual solution in reasonable time.

Finally, when using the mechanism with heuristics, we varied the parameters of the particular heuristic should they have one. The heuristics that can be parametrised are the distance-based heuristic and the GreedyBnB heuristic. For the distance-based heuristic one can use a different importance threshold φ as described in Section 7.1. In particular, we show the performance of different values for this parameter in Sections 8.2, 8.3, and 8.4. These values were $\varphi = \{0.5, 1, 2, 3, 4\}$. Higher values for the importance threshold did not remove enough action vectors and took approximately the same time as the mechanism without heuristics, while lower values for the importance threshold had too much utility loss. Regarding the GreedyBnB heuristic, it can be interrupted before it ends but will still provide a valid solution if interrupted as described in Section 7.3 — though it is expected to produce increasingly good solutions the more time they are given to run. In section 8.5, we illustrate the performance achieved with the heuristic at various cutoff points (after 30, 50, 100, 200, and 500 ms). More cutoff points were considered, but these were the most representative ones to be shown in the figures.

8.2. Number of Action Vectors

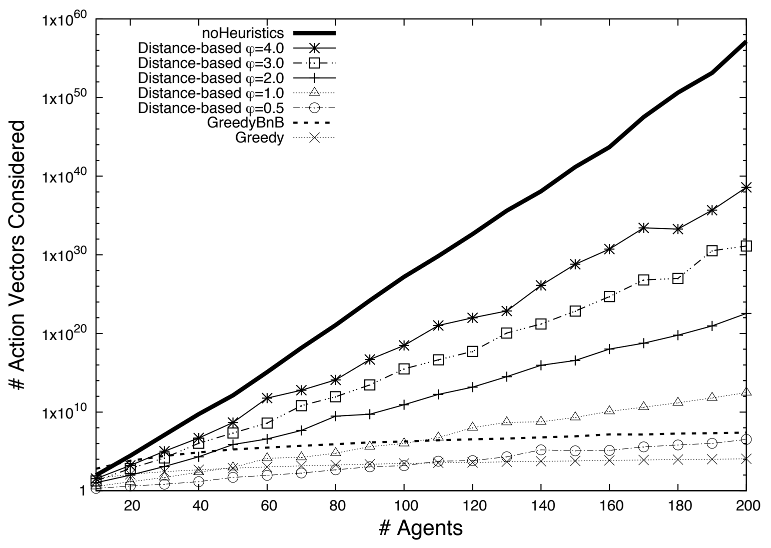

Fig. 2. Average number of action vectors considered per number of agents. Note that Y-axis uses a logarithmic scale.

We measured the number of action vectors to come up with a reasonable estimate of expected execution time, because for more than 40 target agents the experiments where heuristics were not used would not finish after 7 days running. This indicated that without heuristic pruning the negotiation mechanism is not usable in practice.

Figure 2 shows the average number of action vectors (including partial action vectors when using the heuristics) that each approach would need to consider to solve the problem given a number of agents. The plot also shows that the lower φ for the distance-based heuristic, the lower the number of action vectors considered. This is because for low φ values more actions are detected as being more important for one negotiating agent than for the other, so fewer action vectors are generated. We can also observe that both Greedy and GreedyBnB perform similarly, though Greedy considers fewer action vectors. Obviously, GreedyBnB requires more action vectors because it needs to invoke the Greedy heuristics several times to compute a solution.

8.3. Quality of Solutions

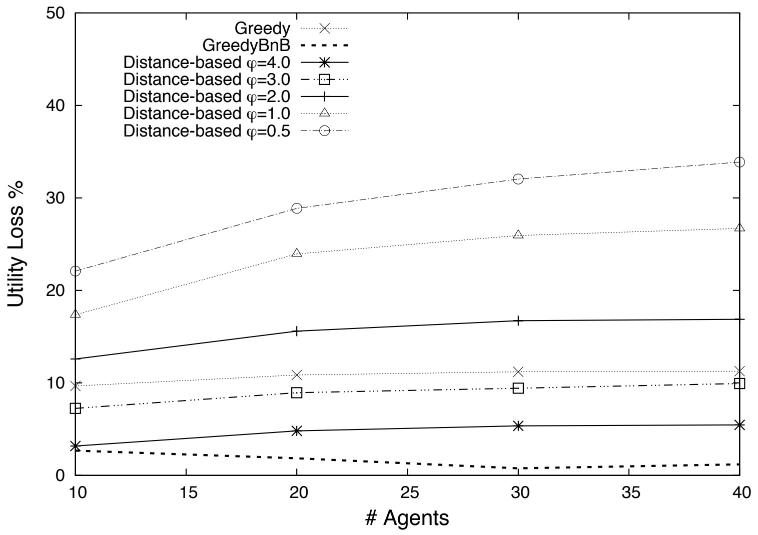

Fig. 3. Average % of utility (product) loss for different numbers of target agents. As before, we were only able to obtain results for up to 40 target agents without heuristics.

In this section, we analyse the differences between the mechanism with and without heuristics in terms of the quality of the deals achieved. Note that the quality of the solutions achieved also tells us the quality of the action vectors that were discarded by the mechanism with the heuristics recall solutions are just action vectors, so if the solutions achieved are of less quality than the ones without heuristics, then this means that the heuristics mistakenly discarded action vectors that were of high quality. In particular, we analysed the quality of the solution achieved considering 3 main metrics: i) the percentage loss in the product of utilities when using the mechanism with heuristics with respect to the mechanism without heuristics; ii) the raw product of utilities for all approaches (mechanism with and without heuristics); and iii) the minimum utility all users get when a solution is found for all approaches (mechanism with and without heuristics).

Figure 3 shows the percentage of utility lost when using heuristics compared to the optimal solution. All heuristics have an impact on optimality, but the best results are obtained with the GreedyBnB heuristic and the distance-based heuristic with $\varphi = 4$. However, if we consider the number of action vectors considered before returning a solution (as shown in Figure 2), the GreedyBnB heuristic is the one that generally offers the best tradeoff between number of action vectors and utility loss. Moreover,

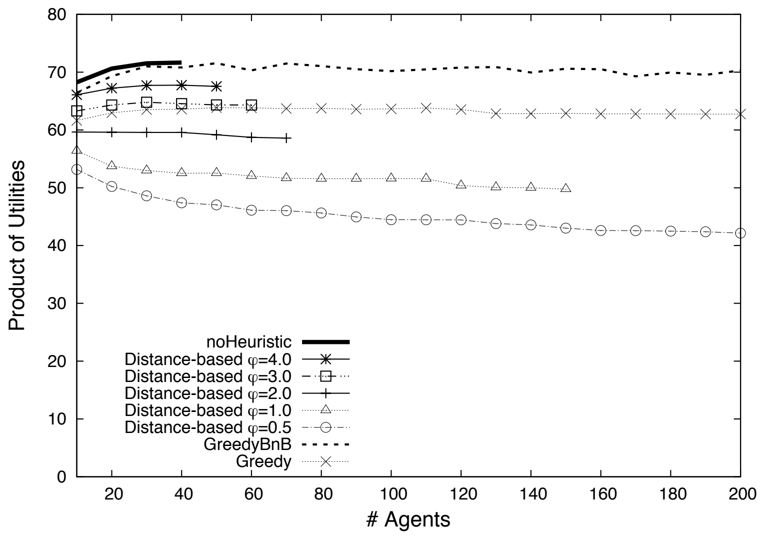

Fig. 4. Raw utility product for different numbers of target agents, heuristic and non-heuristic approaches.

in order to determine how well the heuristics scale when increasing the number of target agents, Figure 4 shows the total product of utilities achieved by each approach. For $\varphi \in \{1, 2, 3, 4\}$ we had the same problem as without use of any heuristic, i.e., we could not obtain results in reasonable time for more than a small number of target agents. In contrast to this, the Greedy and GreedyBnB heuristics scale by far better and clearly outperform the distance-based heuristic with $\varphi = 0.5$, which prunes enough action vectors to produce results in reasonable time for all numbers of target agents considered. We can also observe that when the number of targets increases, GreedyBnB loses less percent of utility with respect to the optimal one. Finally, we can also observe that GreedyBnB clearly outperforms Greedy in terms of utility. Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not exhaustively consider the whole space, and may make commitments to certain choices too early, which prevent them from finding the best overall solution later. Using GreedyBnB, we consider branches of the search space that were initially discarded by Greedy which usually leads to finding a better solution. However, exploring more branches of the search space also comes at a cost — as we have seen previously, GreedyBnB obviously needs to consider many more action vectors than Greedy. In the following section, we assess the difference between these two in terms of actual execution time.

Finally, we sought to assess the quality of the results obtained in terms of the extent individual privacy preferences were being covered. To this aim, we also considered what is the minimum utility achieved among negotiating agents for each deal, i.e., which is the utility for the negotiating agent that is least *favoured* in the negotiation. Figure 5 shows the minimum utility achieved among negotiating agents for each deal considering different numbers of target agents, and heuristic and non-heuristic approaches. We can see that, without heuristics, the negotiating agent that ends up with the lowest utility will always have a utility of at least 7.5. Considering the utility function (Definition 8) and the maximum intimacy value $\mathcal{Y}$ of 10, the maximum utility that negotiating agents can achieve is 10, though this would only be for the case that the deal chosen is the one that favours them the most, which is not possible as compromises are usually done to achieve an agreement. Therefore, without heuristics the

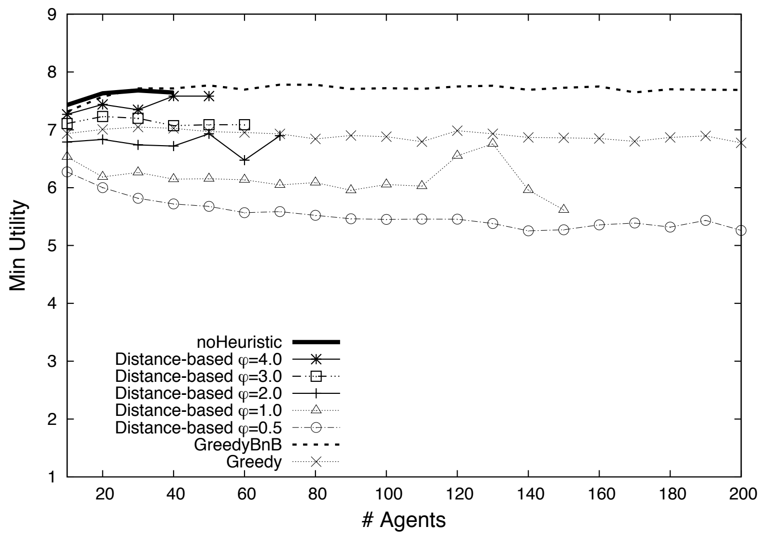

Fig. 5. Minimum utility for different numbers of target agents, heuristic and non-heuristic approaches.

least favoured negotiating agent has always at least 75% of their preferences covered — in other words, it will only lose at most 25% in utility. Moreover, this increases to $\approx$ 78% as the number of target agents increases, because there are more opportunities to find better compromises. Regarding the use of the mechanism with heuristics, the GreedyBnB is again the best heuristic, so that the mechanism with that heuristic provides roughly the same values as the mechanism without heuristics.

8.4. Execution Time

As we were able to obtain results in reasonable time for all the possible configurations when Greedy and GreedyBnB heuristics were used, we were able to compare them in terms of execution time to complement the results obtained in terms of number of action vectors needed to compute a solution. We obviated the distance-based heuristic here because we were only able to obtain results in reasonable time with $\varphi = 0.5$, but with this φ value, the heuristic produced results that are very far from the optimal (up to more than 30%).

Figure 4 shows execution times for the Greedy and GreedyBnB heuristics. We can observe that, from 120 target agents upwards, the GreedyBnB heuristic takes more than ≈ 100 seconds to compute a solution while the Greedy heuristic needs less than $30ms$. Clearly, adding 100 seconds to the process of posting an item could be considers too much for users of social media. However, the Greedy heuristic, which would be fast enough, usually implies a loss of utility of around 10%. Thus, use of any of these two heuristics might be ultimately unsuitable for implementation in a practical tool to resolve actual privacy policy conflicts in social media. Can we find a method that lies somewhere "between" these two that better balances the tradeoff between execution time and optimality?

8.5. Exploiting GreedyBnB anytime capabilities

To come up with such a method, we repeated the experiments, this time exploiting the anytime capabilities of GreedyBnB. In particular, we limited the available computation time to a specific bound as suggested in section 7, i.e., we stopped looking for further possible solutions after a given amount of time elapsed.

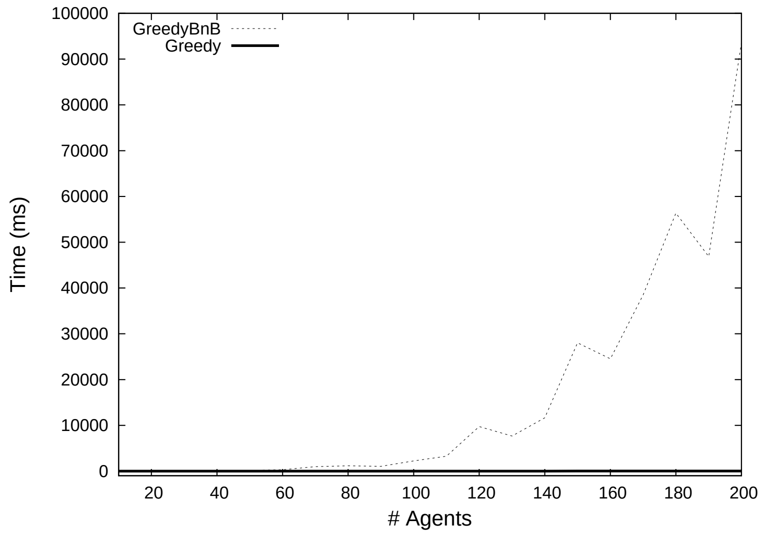

Fig. 6. Execution time for Greedy and GreedyBnB heuristics.

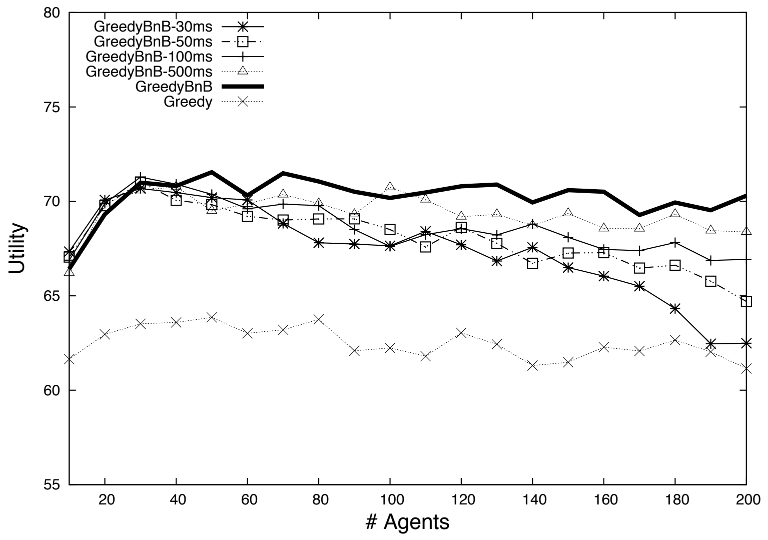

Fig. 7. Utility considering different time bounds for GreedyBnB heuristic.

Figure 5 shows the utility achieved for some temporal bounds compared to that obtained using the unbounded version of GreedyBnB. It also shows the results for the Greedy heuristic as a lower bound for the utility achieved. As expected, the more time GreedyBnB is given, the better the results obtained. Importantly, we can see that with 500ms the results obtained would be very similar to unbounded GreedyBnB, and much better than those achieved using the Greedy heuristic. For instance, for 200 agents the difference between running unbounded GreedyBnB (with a runtime of 100 seconds) and its bounded version limited to 500ms is less than 3%.

8.6. Discussion

The main conclusion from our experiments is that the negotiation mechanism cannot be used in practice for realistic numbers of target agents without the complexity reduction provided by our heuristics. For 200 target agents, for example, we would need to consider $\approx 10^{50}$ action vectors. Moreover, although for a limited number of target agents the distance-based heuristic performs well for some parameter choices, the greedy-based heuristics generally outperforms it both in terms of complexity reduction and in terms of loss of utility, at least for larger numbers of agents. In particular, using the greedy heuristic seems to achieve very good results regarding the number of action vectors considered and execution time — for 200 agents it would consider only $\approx 10^3$ action vectors and would take 30ms, for example — while incurring a loss of optimality that could be acceptable: the heuristics would allow us to reach agreement with 200 target agents while sacrificing only around 10% of utility compared to the non-heuristic version of the algorithm. Using the GreedyBnB heuristic achieves outstanding results in terms of loss of optimality but is worse than Greedy in terms of the number of action vectors considered and execution time required. However, choosing the appropriate parameters to bound the execution time of GreedyBnB seems to achieve the best tradeoff between optimality and execution time. For instance, for 200 agents the bounded version of GreedyBnB would sacrifice only around 3% of the utility obtained while taking only 500ms, which seems to be fast enough for actual use in common social media applications.

9. RELATED WORK

Over the last few years, many studies have been devoted to improving user privacy in social media. Until now, these mechanisms have often been shown not to effectively protect privacy and sensitive information [Zheleva and Getoor 2009; Madejski et al. 2011]. To address this problem, many approaches have recently emerged, e.g. [Carminati et al. 2009; Fong 2011; Fang and LeFevre 2010; Ali-Eldin and van den Berg 2014]. In particular, AI and Multi-agent Systems approaches have been suggested as appropriate to control socio-technical systems like social media [Singh 2013] and as a foundation for social computing[Rovatsos 2014]. Such et al. [Such et al. 2014] present an extensive review of agent and Multi-agent Systems approaches that deal with preserving privacy. An example of these kind of approaches to manage privacy in Social Media is the work of Krupa and Vercouter [Krupa and Vercouter 2012; Ciortea et al. 2012], which proposes an agent-based framework to control information flows in Social Networks. Another example is the work of Kökciyan and Yolum [Kökciyan and Yolum 2014], which proposed an agent-based framework for privacy management and detection of violations based on commitments. Finally, Such et al. [Such et al. 2012] proposed the first agent-based mechanism to decide whether personal information is shared and with whom based on information-theoretic measures of intimacy and privacy . However, all of these approaches do not consider the problem of items that may affect more than one user, and do not support multiple users agreeing on to whom these items are shared. Therefore, these approaches do not consider the privacy preferences of all of the users affected by an item when deciding to whom and whether or not information is shared.

Few works have actually been proposed to deal with the problem of collaboratively defining privacy policies for shared items between two or more users of a social media site. We shall discuss them and how they relate to our work in the following paragraphs.

Wishart et al. [Wishart et al. 2010] propose a method to define privacy policies collaboratively. Their approach is based on a collaborative definition of privacy policies in which all of the parties involved can define strong and weak preferences. They define a privacy language to specify users' preferences in the form of strong and weak conditions, and they detect privacy conflicts based on them. However, this approach does not involve any automated method to resolve conflicts, only some suggestions that users may want to consider when they try to resolve such conflicts manually.

The work described in [Squicciarini et al. 2009] is based on an incentive mechanism where users are rewarded with a quantity of numeraire each time they share information or acknowledge other users (called co-owners) who are affected by the same item. When there are conflicts among co-owners' policies, the use of the Clark Tax mechanism is suggested, where users can spend their numeraire bidding for the policy that is best for them. As stated in [Hu et al. 2011], the usability of this approach may be limited, because users could have difficulties in comprehending the mechanism and specify appropriate bid values in auctions. Moreover, the auction process adopted in their approach implies that only the winning bid determines who will be able to access the data, instead of accommodating all stakeholders' privacy preferences.

In [Hu et al. 2011], users must manually define their *trust* to other users, the sensitivity that each of the items has for them, and their general privacy concern. Then, the authors use these parameters to calculate two main measures, privacy risk and sharing loss. In particular, they calculate the privacy risk and the sharing loss on what they call segments — in our terminology, a segment equals the set of agents in conflict — as a whole, i.e. all of the agents in these segments are assigned the action preferred by either one party or the other in the negotiation. That is, in our terminology only two action vectors — $\vec{v}$ and $\vec{w}$ induced by the privacy policies P_a and P_b respectively — are considered, and the action vector chosen is the one that maximises the tradeoff between privacy risk and sharing loss. Clearly, not considering other possible action vectors could lead to outcomes that are far from optimal.

Finally, there are also related approaches based on voting in the literature [Carminati and Ferrari 2011; Thomas et al. 2010; Hu et al. 2012]. In these cases, a third party collects the decision to be taken (granting/denying) for a particular friend from each party. Then, the authors propose to aggregate a final decision based on one voting rule (majority, veto, etc.). However, the rule to be applied is either fixed [Carminati and Ferrari 2011; Thomas et al. 2010] or is chosen by the user that uploads the item [Hu et al. 2012]. The problem with this is that the solution to the conflicts then becomes a unilateral decision (being taken by a third-party or by the user that uploads the item) and, thus, there is no room for users to actually negotiate and achieve compromise themselves. Moreover, in the latter case, it might actually be quite difficult for the user that uploads the item to anticipate which voting rule would produce the best result without knowing the preferences of the other users. The work presented in [Such and Criado 2014] provides an improvement over these fixed ways of aggregating privacy preferences from users by suggesting 3 different methods that would be selected depending on the particular situation, but again, only a limited number of aggregation methods is considered.

10. CONCLUSIONS

We presented an automated method for detecting and resolving privacy policy conflicts in social media applications. To resolve conflicts, we proposed the use of an automated negotiation mechanism. This mechanism is based on the intimacy among agents, which determines the utility of other agents' proposals. In using intimacy as the determining factor for utility, we followed the findings of most empirical studies with real users, which confirm that it is the main factor that influences their behaviour with regard to setting privacy policies. This suggests that our approach is sound in terms of modelling real users' preferences.

Moreover, in order to reduce the complexity of the negotiation mechanism proposed, we proposed three heuristics and showed through an experimental evaluation comparing the performance of the negotiation mechanism proposed with and without heuristics that: (i) use of the negotiation mechanism is not practicable without heuristics; (ii) the distance-based heuristic only performs well for a limited number of target agents; (iii) the greedy heuristic offers good tradeoffs between complexity and optimality when scaling up the number of target agents; and (iv) the best heuristic overall is GreedyBnB with a time bound. In particular, GreedyBnB bounded to 500ms would sacrifice only 3% in optimality in our experiments.

The research presented in this article is a stepping stone towards automated privacy policy negotiation. We plan to design and implement a social media app with our proposed conflict detection and resolution method, combining it with other existing tools to elicit intimacy and relationship types such as [Fogués et al. 2014]. An interesting future line of research would be to consider the fact that disclosing items can also make relationships evolve [Such et al. 2012], which could play a role in shaping users' preferences about disclosure and negotiation outcomes. Finally, we would also like to extend our mechanism in order to consider the intimacy between the negotiating parties, which is known to influence negotiation strategies and, in particular, may determine to what extent negotiating parties are willing to concede during a negotiation [Sierra and Debenham 2007].

REFERENCES

Amr Ali-Eldin and Jan van den Berg. 2014. A Self-disclosure Framework for Social Mobile Applications. In *Proceedings of the 2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 1–5.

Joseph Bonneau and Sören Preibusch. 2010. The privacy jungle: On the market for data protection in social networks. In *Economics of information security and privacy*. Springer, 121–167.

Barbara Carminati and Elena Ferrari. 2011. Collaborative access control in on-line social networks. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*. IEEE, 231–240.

Barbara Carminati, Elena Ferrari, and Andrea Perego. 2009. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security (TISSEC)* 13, 1 (2009), 6.

Andrei Ciortea, Yann Krupa, and Laurent Vercouter. 2012. Designing privacy-aware social networks: a multi-agent approach. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*. ACM, 1–8.

Lorrie Cranor and Simson Garfinkel. 2005. *Security and usability: designing secure systems that people can use*. O'Reilly Media, Incorporated.

George Danezis. 2009. Inferring privacy policies for social networking services. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*. ACM, 5–10.

Steve Duck. 2007. *Human relationships*. SAGE Publications Limited.

Ulle Endriss. 2006. Monotonic concession protocols for multilateral negotiation. In *AAMAS*. ACM, 392–399.

Lujun Fang and Kristen LeFevre. 2010. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World Wide Web (WWW)*. ACM, 351–360.

Ricard Fogues, Jose M Such, Agustin Espinosa, and Ana Garcia-Fornes. 2015. Open Challenges in Relationship-Based Privacy Mechanisms for Social Network Services. *International Journal of Human-Computer Interaction* 31, 5 (2015), 350–370.

Ricard L. Fogués, Jose M. Such, Agustin Espinosa, and Ana Garcia-Fornes. 2014. BFF: A tool for eliciting tie strength and user communities in social networking services. *Information Systems Frontiers* 16, 2 (2014), 225–237. DOI:http://dx.doi.org/10.1007/s10796-013-9453-6

Philip W.L. Fong. 2011. Relationship-based access control: protection model and policy language. In *Proceedings of the first ACM conference on Data and application security and privacy (CODASPY)*. ACM, 191–202.

Carrie Gates. 2007. Access control requirements for web 2.0 security and privacy. *IEEE Web* 2, 0 (2007).

Eric Gilbert. 2012. Predicting tie strength in a new medium. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1047–1056. DOI:http://dx.doi.org/10.1145/2145204.2145360

Eric Gilbert and Karrie Karahalios. 2009. Predicting tie strength with social media. In *Proceedings of the 27th international conference on Human factors in computing systems (CHI)*. ACM, 211–220.

Mark Granovetter. 1973. The strength of weak ties. *American journal of sociology* 78, 6 (1973), 1360–1380.

Kathryn Green, Valerian J. Derlega, and Alicia Mathews. 2006. Self-Disclosure in Personal Relationships. In *The Cambridge Handbook of Personal Relationships*. Cambridge University Press, 409–427.

Ralph Gross and Alessandro Acquisti. 2005. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society (WPES)*. ACM, 71–80.

Tad Hogg, Dennis M Wilkinson, Gabor Szabo, and Michael J Brzozowski. 2008. Multiple Relationship Types in Online Communities and Social Networks.. In *AAAI Spring Symposium: Social Information Processing*. 30–35.

David J Houghton and Adam N Joinson. 2010. Privacy, social network sites, and social relations. *Journal of Technology in Human Services* 28, 1-2 (2010), 74–94.

Hongxin Hu, G Ahn, and Jan Jorgensen. 2012. Multiparty access control for online social networks: model and mechanisms. *IEEE Transactions on Knowledge and Data Engineering* (2012).

Hongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. 2011. Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC)*. ACM, New York, NY, USA, 103–112. DOI:http://dx.doi.org/10.1145/2076732.2076747

Nadin Kökciyan and Pınar Yolum. 2014. Commitment-Based Privacy Management in Online Social Networks. In *First International Workshop on the Multiagent Foundations of Social Computing (MFSC@AAMAS-2014)*. 1–12.

Yann Krupa and Laurent Vercouter. 2012. Handling privacy as contextual integrity in decentralized virtual communities: The PrivaCIAS framework. *Web Intelligence and Agent Systems* 10, 1 (2012), 105–116.

Fernando Lopes, Michael Wooldridge, and Augusto Q. Novais. 2008. Negotiation among autonomous computational agents: principles, analysis and challenges. *Artificial Intelligence Review* 29, 1 (2008), 1–44.

Michelle Madejski, Maritza Johnson, and Steven Bellovin. 2011. *The Failure of Online Social Network Privacy Settings*. Technical Report CUCS-010-11. Columbia University. 1–20 pages.

Martin J Osborne and Ariel Rubinstein. 1990. *Bargaining and markets*. Vol. 34. Academic press San Diego.

Daniele Quercia, Renaud Lambiotte, Michal Kosinski, David Stillwell, and Jon Crowcroft. 2012. The personality of popular facebook users. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW'12)*. 955–964.

Elie Raad, Richard Chbeir, and Albert Dipanda. 2013. Discovering relationship types between users using profiles and shared photos in a social network. *Multimedia Tools and Applications* 64, 1 (2013), 141–170. DOI:http://dx.doi.org/10.1007/s11042-011-0853-7

Jeffrey S. Rosenschein and Gilad Zlotkin. 1994. *Rules of encounter: designing conventions for automated negotiation among computers*. MIT Press. http://books.google.com/books?id=4ZhPMk3ftpoC

Michael Rovatsos. 2014. Multiagent systems for social computation. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*. IFAAMAS, 1165–1168.

Carles Sierra and John Debenham. 2007. The LOGIC negotiation model. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 1–8.

Munindar P Singh. 2013. Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 1 (2013), 21.

Anna Cinzia Squicciarini, Mohamed Shehab, and Federica Paci. 2009. Collective privacy management in social networks. In *Proceedings of the 18th international conference on World wide web (WWW)*. ACM, 521–530.

Anna Cinzia Squicciarini, Smitha Sundareswaran, Dan Lin, and Josh Wede. 2011. A3P: adaptive policy prediction for shared images over popular content sharing sites. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia (Hypertext)*. 261–270.

Jessica Staddon, David Huffaker, Larkin Brown, and Aaron Sedley. 2012. Are privacy concerns a turn-off?: engagement and privacy in social networks. In *Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS)*. ACM, 1–13.

Lior J Strahilevitz. 2005. A social networks theory of privacy. In *American Law & Economics Association Annual Meetings*. bepress, 42.

Fred Stutzman, Ralph Gross, and Alessandro Acquisti. 2013. Silent Listeners: The Evolution of Privacy and Disclosure on Facebook. *Journal of Privacy and Confidentiality* 4, 2 (2013), 2.

Jose M Such and Natalia Criado. 2014. Adaptive Conflict Resolution Mechanism for Multi-party Privacy Management in Social Media. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 69–72.

Jose M. Such, Agustín Espinosa, and Ana García-Fornes. 2014. A Survey of Privacy in Multi-agent Systems. *Knowledge Engineering Review* 29 (2014), 313–344. Issue 03.

Jose M. Such, Agustin Espinosa, Ana García-Fornes, and C. Sierra. 2012. Self-disclosure Decision Making based on Intimacy and Privacy. *Information Sciences* 211 (2012), 93–111.

Kurt Thomas, Chris Grier, and David M Nicol. 2010. unfriendly: Multi-party privacy risks in social networks. In *Privacy Enhancing Technologies*. Springer, 236–252.

Jose M. Vidal. 2010. *Fundamentals of multiagent systems with NetLogo examples.*

Sami Vihavainen, Airi Lampinen, Antti Oulasvirta, Suvi Silfverberg, and Asko Lehmuskallio. 2014. the clash between Privacy and automation in Social Media. *Pervasive Computing, IEEE* 13, 1 (2014), 56–63.

Yonggang Wang, Ennan Zhai, Eng Keong Lua, Jianbin Hu, and Zhong Chen. 2012. iSac: Intimacy Based Access Control for Social Network Sites. In *Proceedings of the 2012 9th International Conference on Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC)*. IEEE, 517–524.

Douglas R. White and Michael Houseman. 2002. The navigability of strong ties: Small worlds, tie strength, and network topology. *Complexity* 8, 1 (2002), 72–81.

Jason Wiese, Patrick Gage Kelley, Lorrie Faith Cranor, Laura Dabbish, Jason I. Hong, and John Zimmerman. 2011. Are you close with me? Are you nearby? Investigating social groups, closeness, and willingness to share. In *Proceedings of the 13th international conference on Ubiquitous computing (UbiComp)*. ACM, 197–206.

Ryan Wishart, Domenico Corapi, Srdjan Marinovic, and Morris Sloman. 2010. Collaborative privacy policy authoring in a social networking context. In *Proceedings of the 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*. IEEE, 1–8.

Pamela Wisniewski, Heather Lipford, and David Wilson. 2012. Fighting for my space: Coping mechanisms for SNS boundary regulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 609–618.

Elena Zheleva and Lise Getoor. 2009. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web (WWW)*. ACM, New York, NY, USA, 531–540. DOI:http://dx.doi.org/10.1145/1526709.1526781