# Middleware for Pervasive Spaces: Balancing Privacy and Utility

Daniel Massaguer, Bijit Hore, Mamadou H. Diallo, Sharad Mehrotra, and Nalini Venkatasubramanian

Donald Bren School of Information and Computer Science
University of California, Irvine
{dmassagu, bhore, mamadoud, sharad, nalini}@ics.uci.edu

**Abstract.** Middleware for pervasive spaces has to meet conflicting requirements. It has to both maximize the utility of the information exposed and ensure that this information does not violate users' privacy. In order to resolve these conflicts, we propose a framework grounded in utility theory where users dynamically control the level of disclosure about their information. We begin by providing appropriate definitions of privacy and utility for the type of applications that would support collaborative work in an office environment—current definitions of privacy and anonymity do not apply in this context. We propose a distributed solution that, given a user's background knowledge, maximizes the utility of the information being disclosed to information recipients while meeting the privacy requirements of users. We implement our solution in the context of a real pervasive space middleware and provide experiments that demonstrate its behaviour.

## 1 Introduction

Large and dense sensing, communications, and computing infrastructures are enabling the creation of pervasive spaces that offer new possibilities, conveniences and functionalities. Instrumented pervasive spaces that allow observation of entities enable a rich set of applications ranging from surveillance, situational awareness to collaborative applications. Consider for instance, an office environment—here, collaboration can be greatly enhanced if members of a team know where teammates are, what they are doing, and if they are available for discussions. Unfortunately, while a system that provides this information has the potential to improve efficiencies, it can encroach on the privacy of the target individuals (e.g., Peter wants to find Alice who may not wish to be interrupted). A typical technology solution is to provide opt-in/opt-out mechanisms, where targets disable the capture/release of personalizing information either physically (e.g., Alice turns off localization device) or via suitable access control policies.

We argue that such a binary modality is not sufficient to address the privacy needs of future pervasive space applications—individuals are often willing to make personalizing information available based on the needs and context of the request and requestor (e.g Alice is willing to be interrupted if Peter needs an urgent signature). In this paper, we develop a utility-centric formulation of pervasive applications. Observers requesting information specify the utility of the information and targets (about whom information is being requested) express their privacy needs as a negative utility of releasing that information, e.g. Alice's negative utility of being interrupted and Peter's positive utility of finding Alice.

Fig. 2 illustrates the role of a pervasive system middleware. Given application needs, the system observes the pervasive space by probing sensors and interprets the sensor readings to obtain a useful view of the state of the pervasive space.

The role of the pervasive space middleware is to (a) generate a semantically meaningful view of the pervasive space state (e.g., where people are and what they are doing) while hiding the details of how this state is obtained and (b) determine whether (and in what format) to release information to an observer in the pervasive space by realizing the privacy/utility tradeoffs expressed via privacy/utility policies. This paper focuses on the design of the privacy manager, a key component in such a system.

While the basic idea is straightforward, there are a few complications in developing a generalized and flexible system that can address the privacy/utility tradeoff. First, information can be *inferred* in such a system, without explicit requests. For example, knowledge of associations (e.g. Alice and Mary co-program for a project) can inadvertantly reveal information. In the above case, knowledge that Alice is in the conference room and that Mary is in a meeting regarding the project reveals Mary's location (i.e., the conference room). This is a problem if Mary perceives this additional knowledge as a violation of her privacy. This leads us to our first challenge: *the system must account for inferred information in determining a tradeoff.* Second, information can be represented at different granularities—information can be characterized in a hierachical manner—from least descriptive to most descriptive. In the above request, the system can preserve Mary's location privacy by (a) generalizing Alice's location or (b) hiding information on the nature of Mary's meeting. The latter solution would provide Alice's location but increment the uncertainty of Mary's inferred location, which might preserve Mary's privacy. The system must be able to capture and exploit the natural generalization hierarchy offered by the information revealed instead of completely denying access to the information. Third, the discussion above implicetly assumes that people can specify their privacy and information needs. What those needs are, however, is often not clear. Typical privacy definitions where privacy is a binary concept on top of which statistical guarantees are formulated (e.g., k-anonymity, l-diversity, and the like [35, 29, 30], or those based on differential privacy [18, 28]) do not suffice in our scenario since privacy is no longer a binary concept. Furthermore, specifying these needs is at best cumbersome, cognitively difficult and even unfeasible if we expect users to continuously specify their needs for all possible values, contexts, and users. Realistic mechanisms must be in place to obtain the utility functions.

The goal of this paper is to develop a principled approach and framework that can address the above challenges and enable the privacy-utility balance in pervasive space applications.

**Contributions.** The following are the key contributions of this paper.
1) We develop a model of pervasive spaces to represent the various entities (e.g. users, objects), their static properties (e.g. name), and their dynamic properties (e.g. location, activity) the values of which can be represented at different levels of granularity (Section 2).
2) We model the notion of privacy (for targets), not as a binary concept, but as the negative utility associated with each piece of information, and formulate the problem of maximizing the net utility of the information released by the pervasive space system (to an observer) while avoiding privacy violations due to inference (Section 3).
3) We propose a solution to address the privacy preservation and utility maximization problem based on a distributed simulated annealing algorithm (Section 4).

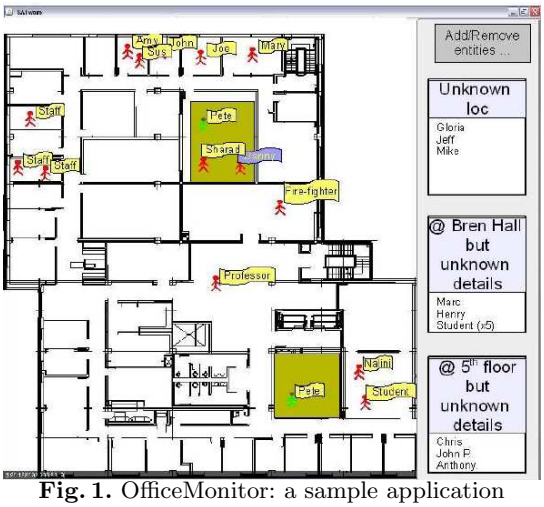


**Fig. 1.** OfficeMonitor: a sample application

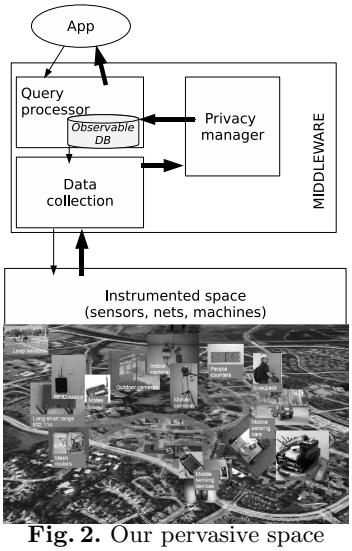


**Fig. 2.** Our pervasive space

4) We extend the existing SATware middleware [21] for pervasive spaces with a privacy manager module (Fig. 2) which incorporates (i) a policy language and mechanisms to express privacy and utility requirements, and (ii) a background knowledge model based on first-order probabilistic datalog clauses and machine learning techniques to populate it, and (iii) the disclosure component that implements the aforementioned simulated annealing-based algorithm (Section 5).
5) We study the performance of our techniques as implemented in a real system (Section 6) by experimenting with scenarios typical from a motivating *OfficeMonitor* application. *OfficeMonitor* is a collaborative application that allows a user (observer) to graphically browse through a university campus map and observe locations and tasks of other people. *OfficeMonitor* also allows targets (users being monitored) to specify rules on *whether* and *how* information about them (e.g., current location, activity) should be released. With this improved awareness, office occupants, for example, are able to prompt their co-workers for impromptu meetings in the most appropriate time

## 2 Pervasive space model as viewed by the applications

From the point of view of the applications, a pervasive space is a physical space in which activities and objects are embedded. In this space, there are 3 types of objects: (1) spatial objects such as rooms, floors, and buildings, (2) people such as Mary, Peter, and Alice, and (3) inanimate objects such as coffee pots, recycle bins, and refrigerators. Each of these objects have attributes such as name, occupancy level, location, salary, level of coffee, and so on. These attributes are either static or dynamic (i.e., they change as a function of time). For instance, name and salary are static whereas location is static for spatial objects but dynamic for people. We call *observable attributes* the subset of attributes that can be sensed by the pervasives space. For example, a pervasive space with video-based people counters and RFID readers can detect both the level of occupancy of a room as well as recognize the people in it.

Our pervasive space middleware will allow applications to view the space as a database whose main table contains the values of the observable attributes over time. The main table has 4 columns: *ObjectId*, *AttributeName*, *AttributeValue*,

and *Time*. We call such a database an observable database ($ODB$), and the main table is called the *Base* table. An example of an ODB.Base table is depicted in Fig. 3.

| ObjectId | AttName | AttValue | Time |
|---|---|---|---|
| Alice | Location | C1/B1/F2/R1 | 10:00:03 *04/24/2009* |
| Alice | Activity | CODING | 10:01:59 *04/24/2009* |
| Mary | Location | C1/B1/F2/R1 | 09:59:42 *04/24/2009* |
| Alice | Location | C1/B1/F2/R1 | 10:00:03 *04/24/2009* |
| Alice | Activity | CODING | 09:59:42 *04/24/2009* |
| Peter | Activity | CODING | 09:59:42 *04/24/2009* |
| John | Activity | COFFEE | 09:59:42 *04/24/2009* |
| John | Location | C1/B1/F2/R1 | 09:59:42 *04/24/2009* |

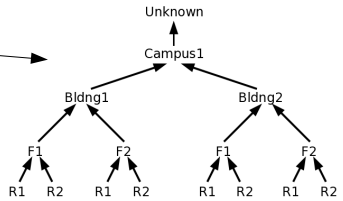


**Fig. 3.** ODB.Base and generalization hierarchy

*OfficeMonitor* type of applications pose continuous queries on ODB.Base. The pervasive space middleware continously answers these queries by deciding which sensors need to be queried and how to interpret the data streams generated by the sensors [21]. At any point of time $t$ the query answers are a set of tuples $Y_{req} = \{< id, att, v, t >\}$ where $id$ identifies an object, $att$ identifies an attribute, and $v$ is the observed value. For the *OfficeMonitor*-type of applications we only need simple filter queries that use selections on the observable attributes such as [1]

SELECT AttributeValue FROM ODB.Base WHERE ObjectId=Alice
AND AttributeName=Location AND Time=Now

We refer to the users who pose these queries as *observers*. We refer to the objects of queries as *targets*. Target identities, attribute values, and time are considered to be generalizable—hierarchies exist to capture these concepts.

### 2.1 Modeling generalization hierarchies

People may be organized into a hierarchy according to their occupation, location may be organized according to physical inclusion, and time may be generalized from seconds to minutes, hours, and days. Given the type of pervasive applications we are interested in enabling (e.g., *OfficeMonitor*), in this paper we only focus on generalizing attribute values. Fig. 3 illustrates a generalization hierarchy for the attribute *Location*. A query for Alice's location may now use this hierarchy to return information at the room, floor, building, or campus level.

We denote generalizations with the $\prec$ partial order and use the notation $x \prec_n y$ to indicate that the minimum number of generalizations between $x$ and $y$ is $n$. For example, <*Mary, Location, Campus1, Now*> $\prec_1$ <*Mary, Location, Campus1/Building1, Now*> and <*Mary, Location, Unknown, Now*> $\prec_2$ <*Mary, Location, Campus1/Building1, Now*>. We extend the definition of $\prec$ to sets of tuples; we say that $X \prec_1 Y$, iff $\exists y \in Y, x \in X$ s.t. $x \prec_1 y$ and $X - \{x\} = Y - \{y\}$; we say that $X \prec_n Y$, iff $\exists\ Z \prec_{n-1} Y$ and $X \prec_1 Z$. We define the partial order $\preceq$ as $x \preceq y$ iff $x = y$ or $x \prec y$, and $X \preceq Y$ iff $X = Y$ or $X \prec Y$. At the top of each attribute generalization hierarchy (i.e., at level 0) there is the null value [35], which we call *Unknown*.

[1] More complex queries (e.g., "select the location and picture of whoever is nearest to the exit"), can be modeled as queries on views that are defined on top of ODB.Base and other tables that contain extra information regarding the space objects, the sensing infrastructure, and so on.

Last, two further assumptions that we make in our model are: (i) we trust the sentient system software and hardware—no privacy leakage is due to them, and (ii) the interest is in the current state: information utility regarding an attribute value decreases exponentially with time.

## 3 Problem Formulation

The task of pervasive space applications such as the OfficeMonitor is to provide answers to users' queries. While the utility of a query response is maximized for the observer when the data is in "the most precise" form, the utility may be quite the opposite for the target of the query. For instance, if location privacy is a concern then revealing accurate information about location is certainly detrimental for the target. There is often such a conflict between the "positive" and "negative" utilities associated with a piece of information that comprises a query response. Traditional access control mechanisms are geared towards deciding between the binary options of *granting* and *denying* access to a piece of information. In contrast, we consider a much larger set of options where the same information is revealed to the observer at a different granularity, i.e. *level of generalization.* For instance, the system may decide to send the tuple <*Mary, Location, Campus1/Building1, Now*> instead of the most accurate version <*Mary, Location, Campus1/Building1/Floor1/Room1, Now*> if it determines (using some criteria which we will describe later) that this is the resolution that achieves the desired degree of tradeoff between privacy of targets and utility of the observer. We claim that our approach allows a greater amount of useful information to be released in general and is acceptable for many pervasive application scenarios where strict access control happens to be too restrictive. Another important feature of our privacy analysis is that we factor in the information disclosed due to inference. The inference problem is especially critical in pervasive spaces where the observer may have substantial amount of background knowledge and historic information (i.e., the contents of ODB.Base over time) using which he can deduce more facts besides what is revealed directly by the response generated by the system. The inference algorithms can lead to substantial increase in load on a real-time system as the knowledge base grows. Therefore, an efficient and scalable implementation is required to deliver a practical solution. In the remainder of this section we describe how we model background knowledge, observer and target utilities, and the information release problem as a constrained optimization problem in an utility theoretic framework. In the next section, we will describe our solution methodology and give efficient algorithms for the optimization problem.

### 3.1 Background Knowledge Model

We model an observer's background knowledge ($BK^{obs}$) as a set of probabilistic first-order Datalog (pDatalog) clauses [25]. pDatalog is much more expressive than propositional logic variations usually used in inference control on statistical databases [30] in that it allows us to model relationships among attributes as well as relationships between attributes and time. With the use of variables, rules can be expressed more concisely. Moreover, pDatalog allows us to reason with probabilities. An example of a rule is:

$$Tuple(Alice, Location, l, t) : p * 0.8 \leftarrow Tuple(Mary, Location, l, t) : p \quad (1)$$

A data element is represented as a multi-attribute tuple of the form *Tuple(objectId, attributeName, attributeValue, time):certainty*. The rule above states that if Mary is at location $l$ at time $t$ with probability $p$, then Alice is also present at the same location (as Mary) with a probability of *0.8*p*. Here *0.8*p* is the *certainty* factor associated with the consequent tuple. The knowledge base consists of rules of the above form along with some other auxiliary information in the form of hierarchies and facts.

The background knowledge base with respect to each observer is the set of rules in the union of a general knowledge base ($KB_G$) (which is common to all observers) and a knowledge base $KB_{Told}$ consisting of the facts that the system has recently revealed to that observer i.e., $KB^{obs} = KB_G \cup KB_{Told}$. The general knowledge base $KB_G$, in turn comprises generalization hierarchies ($KB_{GH}$) as shown in Fig. 3, a knowledge base that expresses the intended usage of the space and its characteristics ($KB_S$), and a set of rules that expresses the individual usage of the space and further attribute and value relationships ($KB_D$) (i.e., $KB_G = KB_{GH} \cup KB_S \cup KB_D$). We call $BK^{sys}$ the union of all observer's background knowledge model, a.k.a. the system's background knowledge. In Section 5.2 we give examples of rules in $KB_D$ and $KB_S$ and describe how the knowledge bases are populated.

### 3.2 Privacy vs Utility

We argue that the potential use (or misuse) of information is what defines the expected utility of information. Given this premise, we derive the definitions of **positive expected utility** of a piece of information for the observer and **negative expected utility** of a piece of information for the target. We formulate our problem as a maximization problem based on these utilities.

Let $Tell(obs, Y_{rel} \subseteq ODB.Base)$ be the action of the system releasing information $Y_{rel}$ to an observer *obs*, where $Y_{rel}$ is a set of tuples of the form $y_r =< id, att, v, t >$ where *id* identifies an object, *att* identifies an attribute, and $v$ is the attribute's value at time $t$. Given that an observer's possible actions are a function of the information he believes in [31, 40, 39], the possible outcomes of $Tell(obs, Y_{rel})$ is defined by the set of actions that the observer could respond with. For example, if due to $Tell(Peter, Y_{rel})$ Peter believes that Mary is in her office, he might go there, and if she is actually in her office, Mary will get interrupted. Furthermore, we assume that the probability of the observer attempting to perform that action is equal to his belief that the piece of information is true.

The outcome of an observer's action has a utility for both the observer and the target of the information released by the system. Namely, it has an immediate positive utility for the observer (Peter gets help from Mary) and an immediate non-positive utility for the target (Mary gets interrupted). We will classify a tuple as *private information* if it has some associated negative utility. Furthermore, since the observer can potentially infer information about other targets (e.g., Alice's location), the observer's might incur a non-positive utility for other targets [2]. The functions $utilityO(obs, y, ctxt^{obs})$ and $utilityT(obs, y, ctxt^{tgt})$ return a number between $[0.0, 1.0)$ and $[-1.0, 0.0]$ respectively, which represent the utility for the observer and the target of an observer learning a specific piece of information $y \in ODB.Base$ in a given *context*. A user's context is defined as a

[2] Utility can also be negative for the observer and positive for the targets [17, 11]. In this paper, however, we focus on a simpler model.

subset of tuples regarding himself and some "benign" objects whose state does not disclose (directly or via inference) private information [3]:

$$ctxt^u = \{y =< id, att, v, t > | id = u \text{ or } id = benignObj, y \in ODB.Base\} \quad (2)$$

We define the observer's *expected* utility for a piece of information as the product of "the probability of the user successfully performing an action (which is equivalent to the observer's belief that a piece of information is true *times* the probability that the information is true)" and "the utility of the outcome of such an action". For notational simplicity, we suppress the *obs* and *context* terms from the expression:

$$EU_O(y) = P(y \mid Y_{rel}, KB_{GH}) * P(y \mid Y_{rel}, BK^{sys}) * utilityO(y) \quad (3)$$

where $P(y \mid Y_{rel}, KB_{GH})$ represents the observer's belief that $y$ is true, and $P(y \mid Y_{rel}, BK^{sys})$ represents the probability that $y$ is true.

We define the target's *expected* utility in a similar manner except for the fact that we have to consider inferences regarding (a) future data and (b) other targets. For example, if Peter knows that Mary joins Alice for dessert, he can infer where Mary might be in the near future if he knows where Alice is having lunch. This way, the target's expected utility can be defined as the product of the probability that the observer will deduce a new piece of information, the information being true, and the (negative) utility for the target[4]:

$$EU_T(y) = P(y \mid Y_{rel}, BK^{obs}) * P(y \mid Y_{rel}, BK^{sys}) * utilityT(y) \quad (4)$$

| Symbol | Description |
|---|---|
| $Y_{req}$ | tuples before discl. control |
| $Y_{rel}$ | tuples after discl. control |
| $Y_{derived}$ | info inferrable from $Y_{rel}$ |
| $GH(Y_{rel})$ | info inferrable from $y_{rel}$ based on gen. hierarchies |
| $Tell(obs, Y_{rel})$ | sentient system's action |
| $\prec$, $\preceq$ | generalization relations |
| $BK^{obs} = KB_G \cup KB_{Told}$ | *obs*'s background knowledge |
| $KB_G = KB_{GH} \cup KB_S \cup KB_D$ | general KB |
| $KB_{GH}$ | generalization hierarchy KB |
| $KB_S$ | intended space usage KB |
| $KB_D$ | domain KB |
| $utilityO$ | obs's utility for a piece of info |
| $utilityT$ | tgt's utility for a piece of info |
| $EU_O$ | Observer's expected utility |
| $EU_T$ | Target's expected utility |

**Table 1.** Symbols

Note that for the target's expected utility we consider the entire observer's background knowledge ($BK^{obs}$) whereas for the observer's expected utility we

[3] In general, there exists no "benign" object since for any given information, theoretically exists some background knowledge that can be applied to obtain some private information [18]. In practice, however, there is information which is more unlikely to allow an observer to infer private information. For example, benign information includes tuples such as *<Campus1/Building1, onFire, true, now>*

[4] Note how this definition is very similar to the disclosure risk definitions of privacy-preserving data publishing [30]. The main difference is that we multiply the disclosure risk by the utility and the probability of a future action happening

consider the generalization knowledge base ($KB_{GH}$). For the targets, we are interested in a worst-case scenario; thus we need to consider any possible leakage of information. For the observer, on the other hand, we are only concerned with the attributes he posed the continuous queries on—i.e., the attributes he is in fact interested in. All the notations are summarized in the Table 1 for easy reference.

Let us denote by $GH(Y_{rel})$ all the information that can be inferred from $Y_{rel}$ given the generalization hierarchy knowledge base $KB_{GH}$. Denote by $Y_{derived}$ all the information that can be inferred from $Y_{rel}$ given the observer's knowledge base $KB^{obs}$. Then, we define the expected utility of $Tell(obs, Y_{rel})$ as the sum of the expected utilities of the data the observer receives as long as these data do not violate the privacy constraints:

$$EU_{Tell(obs,Y_{rel})} = \begin{cases} \sum_{y \in GH(Y_{rel})} EU_O(y) & \text{if } \textit{Private}(Y_{derived}) \\ -\infty & \text{otherwise} \end{cases} \tag{5}$$

where $\textit{Private}(Y_{derived})$ is a boolean function that decides whether privacy is violated or not. In this paper, we take a simple criteria for checking privacy violation. We say that *privacy is met if there is no data whose negative utility is larger than the observer's utility of either that data or any other piece of data that contributed to its inference.* Let us define a minimal independent partition $Y^i_{rel}$ as a subset of $Y_{rel}$ such that no piece of information in $Y^i_{rel}$ allows one to infer a piece of information in $Y_{derived} - Y^i_{rel}$ and vice versa, then privacy is met if:

$$\textit{Private}(Y^i_{rel}) = \begin{cases} \text{true} & \text{if } |EU_T(y_d) * \omega(y_d.t)| \leq EU_O(y_r) \\ & \forall y_r \in GH(Y_{rel^i}), \forall y_d \in Y^i_{derived} \\ \text{false} & \text{otherwise} \end{cases} \tag{6}$$

where $\omega(t) = 2\frac{1}{1+e^{|now - y_d.t|/\tau}}$, with $\tau$ as a small constant (e.g., 1), accounts for information utility decreasing exponentially with time (Section 2).

### 3.3 The Utility Maximization Problem

We cast the problem as a maximization problem where the objective is to find a generalization $Y^i_{rel} \preceq Y^i_{req}$ for each minimum independent partition $Y^i_{rel}$ that maximizes (5) and meets the privacy requirements (6). Namely, the objective is to maximize the observer aggregated expected utility of the information released while ensuring that the largest negative utility of all the information pieces the observer can infer, given his background knowledge and the information being released, is not greater than the largest positive utility of all the information pieces the observer can infer given the generalization hierarchy and the information being released. Formally it can be stated as:

$$\max_{Y^i_{rel}} EU_O(Y^i_{rel}) \tag{7}$$

such that

$$min\_EU_T(Y^i_{rel}) + max\_EU_O(Y^i_{rel}) \geq 0.0 \tag{8}$$

$$Y^i_{rel} \preceq Y^i_{req} \tag{9}$$

where

$$EU_O(Y^i_{rel}) = \sum_{\forall y_r \in GH(Y_{rel^i})} EU_O(y_r)$$
$$min\_EU_T(Y^i_{rel}) = \min_{y_d \in Y^i_{derived}} EU_T(y_d)$$
$$max\_EU_O(Y^i_{rel}) = \max_{y_r \in GH(Y^i_{rel})} EU_O(y_r) \quad (10)$$

# 4 Solution

In this section, we describe how the optimization problem is solved in our system. We utilize the generalization hierarchies to compute a suitable generalization of the tuples before releasing them to the observer. If an observer poses $N$ continuous queries, it is possible that on an event $N$ distinct tuples might need to be generalized. The algorithm therefore has to search for a joint generalization scheme for these $N$ tuples. If there are $m$ levels of generalization per attribute (on average) and $N$ tuples, the number of different generalization schemes is $O(m^N)$. Similar to problems of privacy preservation in data publishing applications, it can be easily shown that this problem is in fact NP-hard for most cases and, hence, efficient polynomial time solutions are unlikely. Now, we describe the properties of the objective function and, based on those, propose a stochastic and distributed scalable algorithm based on simulated annealing to look for an optimal generalization.

## 4.1 Problem characterization

An important property of the objective function (5) is its parallel nature. That is, minimal independent partitions can be solved independently and in parallel. Another important property, is the fact that the utility of a piece of information is never smaller than its generalization, which allows for prunning of the solution space. Formally:

*Property 1.* If $Y^j_{rel}$ is a feasible solution, it is better than any $Y^q_{rel} \prec Y^j_{rel}$.

*Proof.* For any $Y^q_{rel} \prec Y^j_{rel}$, $\sum_{y \in Y^j_{rel}} EU_O(y) = \sum_{y \in Y^q_{rel}} EU_O(y) + \sum_{y \in \{Y^j_{rel} - Y^q_{rel}\}} EU_O(y) \rightarrow \sum_{y \in Y^j_{rel}} EU_O(y) > \sum_{y \in Y^q_{rel}} EU_O(y)$. In other words, the utility of a piece of information is never smaller than its generalization.

Given the exponential size of the feasible region, the need for real-time solutions, the parallel nature of the problem formulation, and the distributed computing affinity of sentient systems, we propose a distributed stochastic solution.

## 4.2 A simulated annealing based solution.

Our solution is based on distributed simulated annealing [26, 32, 34, 27]. Fig. 4 depicts the algorithm. We use the Rete algorithm [19] (an optimized incremental forward-chaining algorithm [34]) on the union of $Y^i_{rel}$ and $BK^{obs}$ to find the minimal independent partitions (*findMinIndPartitions*). That is, every time a rule fires in Rete, all the involved facts along with the rule are joined with every other set that contains any of the involved facts. The time complexity of this first step is polynomial because we make the following assumptions on the background knowledge model. We assume that uncertainty functions ($f$) adhere to the "natural restrictions" [25] of monotonicity ($f(x_1, \ldots, x_n) \leq$

$f(y_1, \ldots, y_n)\ \forall_{i\in[1..n]}\ x_i \leq y_i)$, boundedness $(f(x_1, \ldots, x_n) \leq x_i \forall_{i\in[1..n]})$, and continuity w.r.t its arguments—that is, the higher the premises the higher the consequent, the new data being inferred is as good as its premises, and the certainty function is continuously defined. Since the background knowledge is used to model *possible* privacy violations, resulting facts that are identical except for their associated uncertainty are combined with the MAX function. In the worst-case, the observer will know the rule that resulted in the highest certainty. With these assumptions, the inference analysis terminates in a finite number of steps [25].

Furthermore, the worst-case time complexity of Rete is still linear w.r.t. the number of rules ($r$) and polynomial w.r.t. the number of facts ($f^c$, $c$ = *ruleLength + ruleArity)* as it is for non-probabilistic Datalog clauses [34]. Given that the actions taken for every time a rule is fired have a time complexity of $O(f^3 + 2fr)$, this first part of the algorithm has a worst-case time complexity of $O(rf^c + r(f^3 + 2rf)) = O(rf^c + r^2 f)$.

```
Y_rel = findMinIndPartitions(Y_req, BK^obs)
for each(Y^i_rel ∈ Y_req)
 do n times in parallel
   SimulatedAnnealing(Y^i_rel)
 enddo
endfor
```

**Fig. 4.** Maximization algorithm

Every minimal independent partition $Y^i_{rel}$ is optimized by multiple simultaneous instances of the simulated annealing algorithm from Fig. 5.In this algorithm, a state's neighbor is generated by randomly selecting a tuple and then generalizing it. A neighbouring state is accepted according to the typical acceptance function for simulated annealing $accept(s, T) = e^{-\Delta E/T}$. We define a state's energy $E(Y^j_{rel})$ as:

$$E(Y^j_{rel}) = \rho(\frac{\sum_{y_r \in Y^j_{rel}} EU_O(y_r)}{|Y^j_{rel}|}) + \frac{1}{\rho}(Nat(-\max_{y_r \in Y^j_{rel}}(EU_O(y_r)) - \min_{y_d \in Y^j_{derived}}(EU_T(y_d) * \omega(y_d.t))) \quad (11)$$

```
function SimulatedAnnealing(Y^i_rel)
 Y^j_rel = Y^i_rel.neighbor()
 Y*_rel = max(Y^j_rel, Y^i_rel)
 T = initial Temperature
 while(!terminate)
   if(accept(Y^j_rel, T))
     if(Y^j_rel.energy < Y*_rel.energy)
       Y*_rel = Y^j_rel
     endif
   endif
   if(!change temperature)
     Y^j_rel = Y^j_rel.neighbor()
   else
     T.decrease();
     if(!terminate)
       Y^j_rel = Y^j_rel.neighbor()
     endif
   endif
 endwhile
 return Y*_rel
endfunction
```

**Fig. 5.** Simulated annealing

where $Nat(y)$ returns $y$ if $y >= 0.0$ or 0.0 otherwise, and $\rho = 10^{-r}$, with $r \geq 1$, is the *penalty* associated with violating Constraint 8 . Note that if the initial solution $Y^i_{req}$ is feasible, then $\Delta E(Y^i_{req}) \leq \rho$, which is a number close to 0.0.

We choose the initial temperature $T_0$ to be $\frac{1}{\rho}$ according to the following reasoning. The worst-case $\Delta E$ that we consider is when the neighboring state

violates Constraint 8 by the approximate same amount but utility drops drastically. An upper bound on $\Delta E$ is thus $\Delta E = \rho$. In order to accept this state with a high probability at high temperatures, we set $T_0 = \frac{1}{\rho}$ since $\frac{1}{e^{\rho/T_0}} = 0.99$ for $\rho = 0.1$.

We change the temperature every $\frac{N' * max(m')}{2}$ iterations ($m' < m$ is the maximum number of granularities in $Y^i_{rel}$) because $\frac{N' * max(m')}{2}$ is the average distance from the initial state to the optimal. The temperature schedule follows the typical geometric rule $T_k = \delta * T_{k-1}$. Normally, with such a temperature schedule, $\delta$ is chosen very close to 1.0 such as 0.9 or 0.99 [32]; however, and in a manner similar to [27], since we are running multiple instances of the simulated annealing operator in parallel, we choose a small $\delta$ (e.g, for $r = 1$, $delta = \rho = 0.1$).

The algorithm terminates when a state with energy 0.0 has been found, the temperature reaches $\delta$, or a feasible solution has been found (Property 1). The time complexity of the distributed simulated annealing becomes $O(\log_\delta(\delta/T_0) * \frac{N' * m'}{2} * (rf^c + N)) = O(\log_\delta(\delta * \rho) * \frac{N' * m'}{2} * (rf^c + N)) = O(\log_\delta(\delta^2) * \frac{N' * m'}{2} * (rf^c + N)) = O(Nrf^c + N^2)$. Consequently, the worst-time complexity of the whole maximization algorithm is $O((rf^c + r^2 f) + (Nrf^c + N^2))$—i.e., polynomial w.r.t. the size of the knowledge base and number of queries.

## 5 Implementation

We implemented the Privacy Manager in a real pervasive space composed of the Responsphere infrastructure [4] and the SATware middleware [21]. Together, Responsphere and SATware provide a campus-wide pervasive testbed for interdisciplinary research in situation monitoring and awareness. Responsphere is a pervasive sensing, communications, computing, and storing infrastructure that covers a third of our university campus. It includes more than 200 sensors of different types such as video cameras, RFID readers, networked people counters, and wireless sensor networks (i.e., motes). SATware [21] is a middleware we have developed for executing pervasive applications on top of such an infrastructure. It provides applications with a semantically richer level of abstraction of the physical world compared to raw sensor streams. SATware's processing and programming model is based on operators, which serve as the transition between raw sensor streams to semantically richer information streams. Operators are Java-based mobile agents that implement a simple and data-centric function. For example, SATware provides operators that given a stream of video frames generates a stream of tuples that indicate whether motion has been detected. The Responsphere-SATware framework has been and is being used to test and develop applications such as privacy-preserving video surveillance [22, 7, 43], situational awareness for firefighters (SAFIRE) [6], building visitor tracking [5], technology-induced recycling behaviour, fresh coffee alerts, and others.

In order to enable further applications such as the *OfficeMonitor*, we extended SATware with the Privacy Manager. The high-level design for integrating the Privacy Manager into SATware is shown in Fig. 6, which has 3 key components: (1) Policy Manager, (2) Background Knowledge Generator, and (3) Disclosure Control module.

In the Policy Manager, privacy policies and utilities are specified by users through the Policy Editor, validated by the Policy Processor, and stored into the Privacy DB. The knowledge base representing the background knowledge of users is partly populated by system and space administrators and partly learned

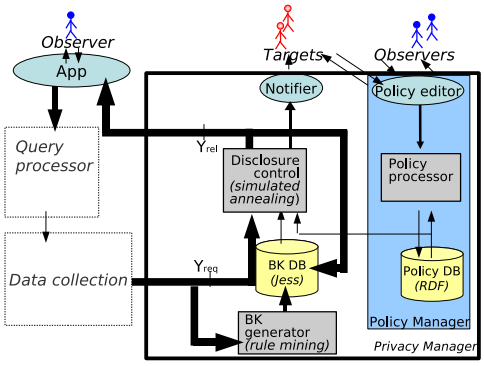


**Fig. 6.** PrivacyManager

| Target | |
|---|---|
| **Labels** | **Utilities** |
| Extremely Sensitive | −1.00 |
| Very Sensitive | −0.75 |
| Sensitive | −0.50 |
| Somewhat Sensitive | −0.25 |
| Not Sensitive | 0.00 |
| **Observer** | |
| **Labels** | **Utilities** |
| Don't Care | 0.00 |
| Information Curiosity | 0.25 |
| Information Useful | 0.50 |
| Information Needed | 0.75 |
| Always Needed | 0.99 |

**Fig. 7.** Utility Scales

(on-the-fly) by the system using the BK Generator, and stored into the BK-DB. Continuous queries are posed by an observer through an application, and their results are transmitted to the Disclosure Control module which analyses the possible information (using the proposed distributed simulated annealing technique) that the observer could infer and, with the active policies, decides which information should be generalized and how. We now describe implementation details (and issues) for the 3 modules.

### 5.1 Policy Manager

We developed the Policy Language for Pervasive Spaces (PLPS) based on the Platform for Privacy Preference (P3P) [16] to assist users (observers and targets) in specifying the privacy policies and utilities (positive for observers and negative for targets). P3P is a W3C standard that enables websites to express their privacy policies in a computer-readable format and provides a protocol to read and process the policies automatically through web browsers. Additionally, it allows web users to express their privacy preferences that can be match with privacy policies specified by the websites. Based on these concepts, PLPS is designed to enable users of pervasive spaces to express their privacy policies to protect their personal information. In PLPS, a policy is defined as a set of statements, where each statement contains: (a) a piece of information; (b) the observer; (c) the retention that defines the length of the observation; (d) the context; and (e) the utility. In addition, each policy is associated with the mandatory elements name of the policy and observer who owns the policy, and the optional elements policy creation date, expiration date, and description. A policy is formally defined as a tuple of the form PP = {PolicyID, Target, CreationDate, ExpirationDate, Statements}, where *Statements* is a set of statements and each statement is formally stated as a tuple of the form Statement = {Observer, Retention, Context, Data, Utility}. The *Retention* element is defined as a tuple of the form {StartDate, EndDate, Frequency}. The *Frequency* element indicates the repetition of the observation defined in the statement and it is drawn from the set {Once, Daily, Weekly, Monthly, Yearly}. An example of a target's privacy policy in the *OfficeMonitor* application could be "Between 01/01/2009 and 03/30/2009, every Monday between 1pm and 4pm, Mary allows the system to tell if she is in her office to her research group members, but not to other users". This example is encoded as follows:

{Policy1, Mary, 01/01/2009, 03/30/2009,

{Group1, [01/05/09-1:00, 01/05/09-4:00, Weekly], Location, C1/B1/F2/R1, 0.0},
{Others, [01/05/09-1:00, 01/05/09-4:00, Weekly], Location, C1/B1/F2/R1, -1.0}}.

PLPS is not only flexible in terms of representing, managing, and interconnecting various types of policies; it also allows the definition of policies at different granularity levels for data attributes. Furthermore, policies are not static but rather the system dynamically updates them as the users specify new needs or tune old ones. We use the Web Ontology Language (OWL) [2] to represent the privacy policy rules modeled in PLPS in the form of ontology and complement it with the Semantic Web Rule Language (SWRL) [3] to express more complex rules. The advantage of using the Semantic Web to represent the policies, is the ability to perform various operations on the policies, including consistency checking, through ontology reasoners. We used the Jena API to implement the module.

**Specifying Utilities:** An important factor in defining the privacy policies is how to obtain the utility values. Obtaining the user's utility function is cognitively difficult [14] and specifying the utility of every possible piece of information for every target, observer, and context is an incredibly tedious task, which can hinder the usability of the approach. To address this issue, we propose a model that allows users to dynamically change their utility values for each policy statement based on their experiences and needs using a graphical continuous sliding scale with 5 labels homogeneously distributed [5]. Fig. 7 shows the 5 labels and the utility values associated with them. Recall that the utility for the observer is in the range $[0.0, 1.0)$ and the utility for the target in the range $[0.0, -1.0]$.

We extended the scale approach by adopting the Conditional Outcome Preference Network (COP-network) [12], for eliciting user preferences and estimating utilities. Applied to our policy model, a COP-network is a directed graph that represents the relative user preferences of the different data. Using this network of preferences and a few utilities "anchored" in some of the labels from Fig. 7, one can estimate the remaining utilities. Three techniques for estimating utilities are included with the COP-network approach, and we have selected and implemented the one that is proven to be more effective: the Longest-Path technique. In short, this technique takes as inputs a COP-network and a set of known "anchored" utilities, selects the longest path of private data in the network for which utilities are unknown, and compute utilities for those private data in a way that the preference ordering in the network is preserved. This process continues until all the private data has been considered.

### 5.2 Background Knowledge Generator

The privacy manager implementation also needs to deal with the issue of populating the knowledge base that represents the background knowledge of the users. In our background knowledge model, we identified three types of background knowledge that we modeled with pDatalog clauses: the generalization hierarchy ($KB_{GH}$), the space intended usage ($KB_S$) such as most people check their emails in their office, and the space individual usage ($KB_D$). We propose populating the knowledge bases as follows. The information in $KB_{GH}$ and $KB_S$ is initially populated by system and space administrators and continuously calibrated by the middleware. Calibration of rules in $KB_{GH}$ and $KB_S$ is done

[5] We used 5 intervals as in the Likert scale, which is a well-accepted psychometric ordinal scale used in questionnaires and survey research [1]. Five levels are the usual choice since 3 do not provide enough variability and 7 offer to many choices

by regularly matching the recent data observed by the system with their rules. Borrowing the terminology from rule-mining algorithms, we call support $s\%$ the number of times a rule's premises appear divided by the number of tuples observed, and we call confidence $c\%$ the percentage of these times that the rule consequent also appears. We update a rule in $KB_{GH}$ or $KB_S$ when $s\%$ is above a threshold and we cannot reject the null hypothesis that the average times the rule holds for all individuals $c\%$ has not changed.

The information in $KB_D$ is not pre-populated; rather, it is learned by the system overtime. Similar to $KB_{GH}$ or $KB_S$, we create an exception rule in $KB_D$ when $s\%$ is above a threshold and we cannot reject the null hypothesis that the average times the rule holds for some individual $i$ ($c_i\%$) and the average times the rule holds for all individuals ($c\%$) are different (e.g., whereas most people check their emails in their office, Peter does it at the conference room). Furthermore, $KB_D$ contains rules learned by regularly running association-rule mining algorithms, such as [9], on the information observed by the system. Given a set of items $I = \{i_0, \ldots, i_k\}$ and a set of transactions/baskets $T = \{t_0, \ldots, t_n \mid t_i \subseteq I\}$, rule association mining algorithms produce propositional rules of the form $Y \leftarrow X$, with $X \subset I$ and $Y \subset I$, where each rule has an associated confidence $c\%$ and support $s\%$. We propose *basketizing* the data observed by the system into time-based baskets at different time granularities and then mining association rules for each granularity. For instance, we would (i) put all the tuples whose time is within the same second in the same basket and then mine for rules such as "Mary writes emails at her office"; (ii) put all the tuples whose time is between 8 a.m. and noon in the same basket and mine for rules such as "Peter has coffee in the mornings"; (iii) put all the tuples whose time is in a Tuesday and derive that "Alice goes to board meetings on Tuesdays"; and so on up to weekly baskets. Note that we can, this way, derive both rules regarding the space usage (e.g, "Mary writes emails at her office") and inter-object relationships (e.g., "Mary's location is the same as Alice's 4/5 times").

We will then *upgrade* the resulting propositional rules to pDatalog rules by using $c\%$ as the uncertainty associated with the rule and fixing $s\%$ as a system parameter. Moreover, whenever a rule appears consistently among entities it will be generalized and added to $KB_S$—for example, if most people have coffee in the morning. The implementation details of the background knowledge learning algorithms are out of the scope of this paper. In here, we limit ourselves to show that the knowledge can be obtained and, hence, assumed that it has.

### 5.3 Disclosure Control

The disclosure control module is the key to the approach. Given a set of base results to an observer's queries, the disclosure control consults the policy database and BK to determine how to release the information without violating the target privacy, while maximizing the observer utility. Fig. 8 depicts the details of the implementation of the disclosure control module. Our solution is implemented as a graph of operators. The $B$ operator outputs a series of sets where each set contains meta-information on an independent component. Namely, each set contains a small knowledge base with the relevant rules for this component and a subset of the tuples in $Y_{req}$. Similar to [15], we extend the Jess *deffact* template with an extra slot for the associated uncertainty and an extra rule to handle the combination of evidences on the same fact. The output of the $B$ operator is forwarded to the scheduling operator, which forwards each input to a different *PSA* operator in a round-robin manner. Each *PSA* operator executes the parallel simulated annealing on the minimum independent components using also the

extended Jess. The utility functions come from the $CTXT$ operators which, depending on the current context, query the policy DB for the active policies. The results of the $PSA$ operators are forwarded to the $\pi$ operators. The $\pi$ operators then filter the data so the $UI$ operators receive the $< id, att, v, t >$ tuples they expected.

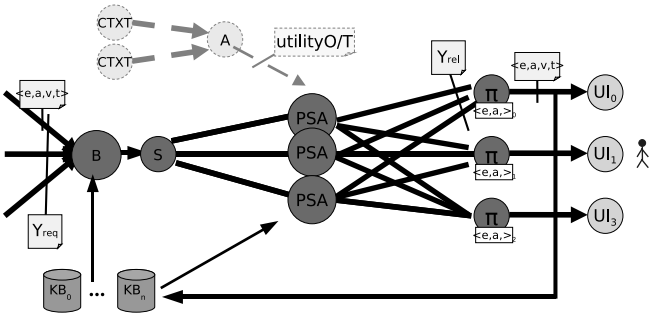


**Fig. 8.** Disclosure control.

## 6 Experiments

The main goal of the disclosure control submodule is to be able (i) to produce good results (high utility with adequate privacy) and (ii) to do so in real-time. To test the parallel simulated annealing based approach (PSA), we compared it to two simpler and centralized approaches: a brute force search (BF) and an anonymity-based approach (minGen). The BF approach was implemented as a depth-first search (DFS) with pruning based on Property 1. The minGen approach was an adaptation of the typical privacy definition in data publishing [37], where data is either private or public and the goal is to guarantee that private data attains a certain degree of anonymity while the information lost due to generalization is minimized. We measured the information lost as the average uncertainty introduced. Formally:

$$\max_{Y_{rel}} \sum_{y_q \in Y_{req}} \frac{P(y_q|Y_{rel}, KB_{GH})}{|Y_{req}|}$$
$$s.t. \forall_{(y_d \in Y_{derived}, y_d \in P)} \; P(y_d|Y_{rel}, BK^{obs}) > (1/k) \tag{12}$$

where $P$ is the set of private data. This algorithm was implemented also with DFS with pruning. For comparison purposes, we considered all data with negative utility to be private and we assumed a fixed value of $k = 4$. We implemented 3 versions of the disclosure control submodule, one for each approach.

The experiments setup was as follows. We used scenarios from the *OfficeMonitor* application to create a realistic experimental setup. For the basic case, we instantiated the *OfficeMonitor* queries for a typical project group (7 people). Queries generated tuples of the form <*id,att,v,t*> which were then routed to the Disclosure Control submodule. The query set generated 9 tuples every second over a period of time (our results are an average of 36 of such runs). The tuples represented 2 different minimal independent partitions. For the first partition, the utility values were set such that the solution was 5 generalizations away

from the initial solution (the distance to generalizing everything to unknown was 28 generalizations). For the second partition, the solution was 0 generalizations away from the initial solution. The knowledge base had 12 facts, and 47 rules (similar to rule (1)), 14 of which were related to the 9 tuples being sent. We believe this to be a typical setup and dimensioning for the *OfficeMonitor* application.

Our first set of experiments compared the PSA approach with the BF and the minGen approaches. In order to further study the characteristics of the PSA approach, we instantiated 5 variations of it. These instantiations differed on the degree of concurrent exploration (i.e., number of threads) being used. We call these PSA($x$) where $x$ is the number of concurrent explorations on the same partition. We then compared PSA(1), PSA(6), PSA(11), PSA(16), and PSA(21) with the BF and minGen versions in terms of time overhead and utility loss. Fig. 9 shows the results of our experiments with a dual-core machine featuring an AMD Turion 64 X2 at 2.0Ghz, with 3GB of memory, and running Linux. While the BF approach always finds the optimal, it takes far more time than the PSA approaches (minutes *versus* a few hundred milliseconds), which manage to return a solution very close to the optimal when incrementing the amount of concurrent exploration. The greatest utility loss is incurred by minGen, as expected (recall from above that minGen can only differentiate between public and private data and thus, it considers all data with negative utility equally private).

Our second set of experiments studied the scalability properties of the PSA approach. Fig. 10 suggests that, whereas our approach still takes a feasible amount of time for 12, 18, 24, and 30 tuples per second, it does not scale linearly but polynomially (which is expected, recall from Section 4.2 that the time-complexity of the disclosure control algorithm is polynomial w.r.t. the number of queries). PSA might not scale for other applications beyond the *OfficeMonitor* where applications need a large amount of tuples per second. However, the scalability of our approach can be improved by making use of (a) the parallel nature of the problem at hand and (b) the very nature of pervasive spaces which allows for distributed computing. In most cases, query results will have several minimum independent partitions in it. Each of these partitions can be optimized separately. Fig. 11 studies the effect of adding more PSA operators. It shows that, since, we had 2 minimum partitions, there is an important reduction of the time needed to find the solution when we had a number of PSA operators close to the number of independent partitions. With one PSA operator, independent partitions get queued in front of the operator; with 2 or more PSA operators, these queues tend to be small or almost empty; however, there is a point where the overhead of having the extra operators starts weighing more than their distributed benefit.

## 7 Related work

Several privacy and anonymity definitions and metrics along with specific solutions have been proposed in the literature. For instance, [35] presents the metric of $k$-anonymity, which expresses the fact that in the worst-case the value of a user's sensitive attribute can only be narrowed to a set of size $k$. In [37], the authors present a framework to achieve $k$-anonymity by generalization and suppression. In [29] the authors extend the $k$-anonymity metric with $l$-diversity which guarantees that it takes at least $l-1$ pieces of negative background knowledge (i.e., "Tom does not have arthrities") to sufficiently disclose *the* sensitive

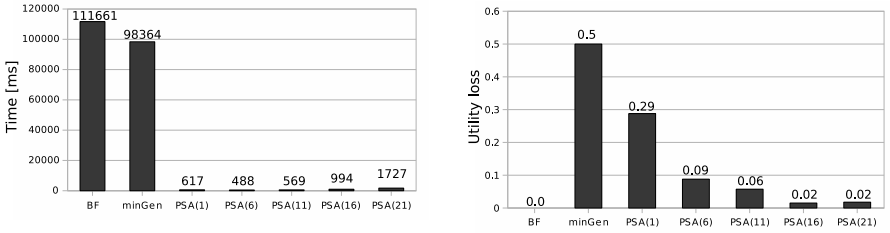


**Fig. 9.** Comparing average time overhead and utility loss of PSA with different concurrent explorations with BF and minGen

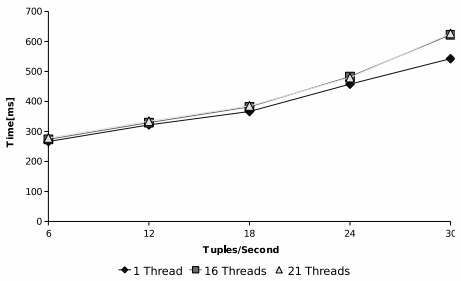


**Fig. 10.** Behaviour of PSA as the number of tuples/sec increases

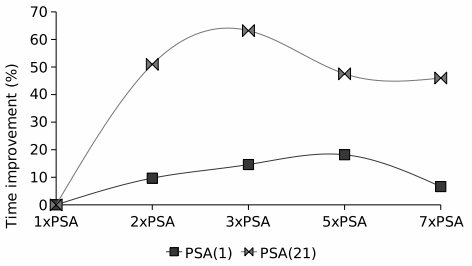


**Fig. 11.** Effect of the increase of number of PSA operators

value of any individual by assuring that the $l$ most frequent sensitive values are approximatly equi-probable. Positive background knowledge regarding *the* sensitive attribute of the type "If Tom has the flu his wife has it as well" is considered in [30], where its authors show the $k$ worst rules that can be in a users' background knowledge and provide polynomial mechanisms to still be able to guarantee a degree of anonymity. In [18] the author proves the impossibility for absolute privacy in statistical databases and defines the alternative metric of *differential privacy*, which is a metric relative to the risk of a user participating in a statistical database. Nonetheless, none of the previous definitions applies to our scenario: we need a non-binary definition regarding information that is not useful in an anonymous manner—the *OfficeMonitor* is not interested in statistical data.

Our work here is similar to QoS-related work in stream systems. Stream systems such as Aurora [13] use semantic shedding [38] as one of the techniques to decide which tuples to drop when resources run low—that is, the less useful the data is for the recipient, the earlier it gets dropped. Here we take this concept further by deciding to drop (or generalize) tuples when a user's privacy would be violated.

Privacy in pervasive spaces has been researched at multiple levels. At the network layer, [10] combines hop-to-hop routing based on handles with limited public-key cryptography to preserve privacy from eavesdroppers and traffic analyzers. At the architectural level, and in a manner similar to outdoor GPS [24], solutions such as Cricket [33] and Place Lab [36] protect a user's (private) location by having a user's carry-on device calculate its location based on a series of beacons from the infrastructure rather than having the infrastructure compute the location as in [42] and [8]. In contrast, we assume that the sensor might not have enough context and resources to compute observations nor it is the final

recipient of information (i.e., it is the system who captures and interprets the information and applications, on behalf of their users, the recipients of information). Other work regarding privacy in pervasive spaces includes the framework for evaluating privacy control and feedback proposed for IMBuddy contextual IM service [23], which strives to improve users understanding of privacy implications through feedbacks. They do not take into account, however, information that can be inferred as a result of the information being disclosed.

Using Semantics Web technologies as means for describing and reasoning about privacy policies in different domains including pervasive environments are becoming common [41, 20]. Relevant to our privacy policy language is the semantic context-aware policy model based on Description Logic (DL) ontologies and Logic Programming (LP) rules in [41]. Central to this approach is the specification of policies based on context rather than the usual way of using roles and identities of users. In user-centric pervasive space applications such as OfficeMonitor, however, identity-based policies are still necessary since privacy is an individual-centric concept.

## 8 Conclusions and Future work

To build a pervasive space middleware that allows applications to query the state of the objects in a given space is indeed a challenging task. One of the main challenges stems from the fact that some of the objects being monitored are people. The middleware needs to make sure that the query answers it provides to the applications do not violate privacy. In this paper, we proposed a novel approach for modeling privacy in the context of pervasive-space-supported collaborative work. We turned away from a traditional binary definition where information is either public or private and proposed a utility-based definition where information is associated with a positive utility for the querier and a negative utility for the target of the query. Moreover, further information that the querier might be able to infer is also associated with a negative utility. With this definition, we proposed a framework where the system has to decide, at every time instant, which information should be generalized and how much, such that privacy is preserved and utility for the querier is maximized.

Our first approach to solve the maximization problem is based on a distributed simulated annealing algorithm. We implemented our approach in an existing pervasive space middleware. To realistically instantiate such an approach, we also had to address the problem of obtaining and representing the utility functions and obtaining and representing a user's background knowledge. We proposed solutions for both problems.

Future directions opened by this paper include considering other types of applications where aggregated information and other mechanisms beyond generalization of attributes are relevant. We did not deal here with queries such as "Select the room with the maximum number of people in it". Privacy on these type of information is of a different nature and has its own challenges—anonymity-based definitions might be more appropriate. Examples of further mechanisms one might want to explore are generalization of identity and time. Identity generalization is specially interesting and, again, of a different nature: a system can only safely generalize "Alice" to "programmer" if the querier is learning information about other $k - 1$ programmers and he cannot tell who they really are. Last, another mechamism would be to trade delay for privacy to avoid time-and-domain based inferences such as inferring that Alice is still in

the building because she was in its top floor two minutes ago—which could be a privacy violation if the context had changed over the last two minutes.

## 9 Acknowledgements

The authors would like to thank the SATware team for their dedication to the project and specially to Roberto Gamboni and Jay Lickfett for his help on mantaining and extending the implementation, Francisco Servant for implementing the first version of the policy management system, Haynes Mathew George for implementing the first prototype of the SATware's query processor, and Ronen Vaisenberg for his advise on the early stages of the paper on how to focus it. We would also like to thank the anonymous reviewers for their helpful comments.

This research has been supported by the National Science Foundation under award numbers 0331707, 0403433, and 0331690.

## References

1. Likert scale. wikipedia, the free encyclopedia.
2. Owl web ontology language guide, February 2004.
3. Swrl: A semantic web rule language combining owl and ruleml, May 2004.
4. Responsphere. http://www.responsphere.org, 2007.
5. RFID Tag lookup. http://www.ics.uci.edu/community/events/openhouse/rfid.php, 2009.
6. SAFIRE. http://www.ics.uci.edu/%7Eprojects/cert/verticals.html, 2009.
7. SATrecorder. http://www.ics.uci.edu/%7Eprojects/SATware, 2009.
8. M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *Computer*, 34(8):50–56, 2001.
9. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
10. J. Al-Muhtadi, R. Campbell, A. Kapadia, M. Mickunas, and S. Yi. Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In *ICDCS*, volume 22, pages 74–83, 2002.
11. K. Anderson and P. Dourish. Situated privaces: Do you know where you mother (trucker) is? In *Proc. HCI International*, 2005.
12. S. Buffett and M. Fleming. Applying a Preference Modeling Structure to User Privacy. *NRC Publication Number: NRC 49372*, 2007.
13. D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams–a new class of data management applications. Technical Report CS-02-04, Brown Computer Science, February 2007.
14. U. Chajewska, D. Koller, and R. Parr. Making Rational Decisions Using Adaptive Utility Elicitation. In *AAAI*, pages 363–369, 2000.
15. D. Corsar, D. Sleeman, A. McKenzie, and U. Aberdeen. Extending Jess to Handle Uncertainty. In *AI-2007*. Springer, 2007.
16. L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The Platform for Privacy Preferences 1.0 (P3P1. 0) Specification. *W3C Recommendation*, 16, 2002.
17. P. Dourish and K. Anderson. Collective information practice: Exploring privacy and security as social and cultural phenomena. *HCI*, 21:319–342, 2006.
18. C. Dwork et al. Differential privacy. *Proc. ICALP*, 2006.
19. C. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *IEEE Computer Society Reprint Collection*, pages 324–341, 1991.
20. G. Hogben. Describing the p3p base data schema using owl. In *A WWW2005 Workshop on Policy Management for the Web*, 2005.
21. B. Hore, H. Jafarpour, R. Jain, S. Ji, D. Massaguer, S. Mehrotra, N. Venkatasubramanian, and U. Westermann. Design and implementation of a middleware for sentient spaces. In *Proceedings of ISI'07*, 2007.

22. B. Hore, J. Wickramasuriya, S. Mehrotra, N. Venkatasubramanian, and D. Massaguer. Privacy-preserving event detection in pervasive spaces. In *PerCom 2009*, 2009.
23. G. Hsieh, K. Tang, W. Low, and J. Hong. Field deployment of IMBuddy: A study of privacy control and feedback mechanisms for contextual im. *Lecture Notes in Computer Science*, 4717:91, 2007.
24. A. Ivan. Getting. The global positioning system. *IEEE Spectrum*, 30(12):36, 1993.
25. M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. *LNCS on ICDT*, 88:102–117, 1988.
26. S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.
27. K. Krishna, K. Ganeshan, and D. J. Ram. Distributed simulated annealing algorithms for job shop scheduling. *IEEE Transactions on Systems, Man. and Cybernetics*, 25(7), July 1995.
28. A. Machanavajjhala, D. Kifer, J. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE 2008*, pages 277–286, 2008.
29. A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity.
30. D. Martin, D. Kifer, A. Machanavajjhala, J. Gehrke, and J. Halpern. Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. In *ICDE*, 2007.
31. D. Massaguer, V. Balasubramanian, S. Mehrotra, and N. Venkatasubramanian. Multi-Agent Simulation of Disaster Response. In *ATDM–AAMAS 2006*, May 2006.
32. D. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1998.
33. N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *MobiComp*, pages 32–43. ACM Press New York, NY, USA, 2000.
34. S. Russel and P. Norvig. *Artificial Intelligence: a modern approach–second edition*. Prentice-Hall, 2003.
35. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1998.
36. B. Schilit, A. LaMarca, G. Borriello, W. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson. Challenge: ubiquitous location-aware computing and the" place lab" initiative. In *WMASH*, pages 29–35. ACM New York, NY, USA, 2003.
37. L. Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *Int'l journal of uncertainty fuzziness and knowledge based systems*, 10(5):571–588, 2002.
38. N. Tatbul, U. Çetintemel, S. B. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *VLDB*, pages 309–320, 2003.
39. L. Thow-Yick. The basic entity model: A Fundamental Theoretical Model of Information and Information Processing. *Information Processing and Management*, 30(5):647–61, 1994.
40. L. Thow-Yick. The basic entity model: A theoretical model of information processing, decision making and information systems. *Information Processing and Management*, 32(4):477–487, 1996.
41. A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In *International Semantic Web Conference*, pages 473–486, 2006.
42. R. Want, A. Hopper, and V. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
43. J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian;. Privacy protecting data collection in media spaces. In *ACM Multimedia 2004*, 2004.