A spatiotemporal model of strategies and counter strategies for location privacy protection

Matt Duckham*, Lars Kulik†, and Athol Birtley†

*Department of Geomatics
University of Melbourne, Victoria, 3010, Australia
mduckham@unimelb.edu.au

†Department of Computer Science and Software Engineering
University of Melbourne, Victoria, 3010, Australia
lars@csse.unimelb.edu.au
a.birtley@ugrad.unimelb.edu.au

Abstract. Safeguarding location privacy is becoming a critical issue in location-based services and location-aware computing generally. Two drawbacks of many previous models of location privacy are: 1) the models only consider a person's location privacy protection, but not the invasion of location privacy by external agents; and 2) the models are static and do not consider the spatiotemporal aspects of movement. We argue that, to be complete, any model of location privacy needs to enable the analysis and identification of techniques both to protect and to invade an individual's location privacy over time. One way to protect an individual's location privacy is to minimize the information revealed about a person's location, termed *obfuscation*. This paper presents an explicitly spatiotemporal model of location privacy that models a third party's limited knowledge of a mobile individual's location. We identify two core strategies that a third party can use to refine its knowledge, so potentially invading that mobile individual's location privacy. A global refinement strategy uses the entire history of knowledge about an agent's location in a single step. A local refinement strategy iteratively constructs refined knowledge over time. We present a formal model of global and local refinement operators, and show how this formal model can be translated into a computational model in a simulation environment.

1 Introduction

Location privacy can be defined as a special type of *information privacy* that concerns the claim of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others [6], cf. [21]. The emergence of low-cost location-aware computing, which combines powerful mobile computing platforms, wireless communication capabilities, and ubiquitous integrated positioning systems, has led to location privacy being acknowledged as a key challenge in information science (e.g., [17]). A failure to safeguard location privacy has been linked to a range of undesirable effects, including unsolicited marketing and *location-based* "*spam*"; decreased *personal safety*, such as might result from stalking or assault; and

intrusive inferences, where other personal data about an individual is inferred from that individual's location over time [11, 14, 19].

This paper presents a new model of location privacy based on a formal analysis of mobile individuals and the knowledge a third party may have about the location of those individuals over time. The approach extends previous work in two significant ways.

- The approach models both the strategies an individual may employ to protect his or her location privacy and the *counter strategies* that a third party might employ to subvert these strategies. Most previous work has focused solely on strategies for location-privacy protection.
- The approach adopts a *spatiotemporal* model of location privacy. This model can be used to account for an individual's movements over time and the refinements a third party may make to its knowledge using such spatiotemporal information.

Following a brief literature review in Section 2, we present the key concepts that underlie this approach to location privacy in Section 3. The formal model, founded on set-based and algebraic techniques, is presented in Section 4. Section 5 discusses the implementation of refinement operators and Section 6 summarizes our formalization of refinement operators.

2 Background

Conventional techniques for protecting (location) privacy can be categorized into three classes: *regulation* [9], *privacy policies* [16, 20], and *anonymity* [8, 10]. Each of these approaches plays an important role in providing a complete solution to location privacy, but also has its limitations. A survey of techniques for location privacy protection is given in [6].

Regulation (such as data protection legislation) and privacy policies are trust-based mechanisms for proscribing unacceptable uses of private location information. Ultimately, any trust-based mechanism is vulnerable to inadvertent or malicious disclosure of private information. Anonymity-based techniques dissociate information about an individual, such as location, from that individual's actual identity [18]. Anonymity also has limitations, especially in spatial application domains. A person's identity can often be inferred from his or her location, making anonymity vulnerable to data mining [1, 7]. Further, using anonymity presents a barrier to authentication and personalization, required for many location-based applications [12, 15].

In [4, 5], the authors presented obfuscation as a fourth type of location privacy protection technique. *Obfuscation* is the process of deliberately degrading the quality of information about a person's location, with the aim of protecting that person's location privacy. Using obfuscation, individuals may choose to reveal varying degrees of information about their location (for example, suburb, block, street, or precise coordinate-level information). In common with anonymity-based techniques, obfuscation aims to hide information. However, obfuscation is an explicitly spatial privacy protection technique that aims to hide information about location rather than identity. Some recent

work has begun to extend privacy policy and anonymity approaches with spatial capabilities (e.g., "spatial cloaking" [10] and the "need-to-know principle" [20]). Obfuscation is complementary to existing privacy protection techniques, but can extend these techniques by enabling greater use of authentication and personalization and by increasing the security of location information from data mining and unauthorized access.

Unlike previous work which has focused primarily on ways to effectively hide location information to protect location privacy, in this paper we are also interested in understanding the ways in which a third party can refine information about an agent's location. The third party could be a hostile agent attempting to invade an individual's privacy or a legitimate security agency attempting to locate an agent more precisely. Our core thesis is that *a complete treatment of location privacy requires an understanding of both the strategies for hiding location information and the counter strategies for revealing hidden location information*.

3 Approach

The approach taken in this paper builds on recent work in [4,5], introduced in the previous section. We assume an agent is moving around a geographic environment, carrying or interacting with some location-aware technology (such as a location-aware navigation system or a simple cell phone). Information about the agent's location is being tracked by third party (3P). Examples of 3Ps might include: co-workers or employers, telecommunications companies, location-based service providers, legitimate security organizations, or jealous spouses. However, either because of technological limitations or because the agent wishes to protect its location privacy, we assume the 3P's knowledge (3PK) of the agent's location is imperfect[1].

The quality of the 3PK may be lowered by a lack of detail (imprecision). For example, location in a conventional cell phone is typically resolved to the level of a single cell, usually an area of between 200m and 2km in diameter. The system can generate highly accurate knowledge about the presence or absence of an agent in a particular cell, but the precise location of agent inside the cell is unknown. The quality of the 3PK may also be lowered by inaccuracy. For example, an agent that wishes to protect its location privacy may deliberately introduce inaccuracies into the location information it reveals to a 3P (one type of obfuscation discussed in [5]). The overall scenario is summarized graphically in Figure 1, where combinations of obfuscation and/or technical limitations give rise to imprecision and/or inaccuracy in the 3PK of an agent's location.

In such a situation, two important questions are:

1. How can a 3P improve its knowledge of the location of an individual (thereby potentially invading an agent's location privacy)?
2. What strategies and spatial behaviors can an agent adopt to counter a 3P's attempts to subvert the agent's location privacy?

[1] Conversely, if the 3P possesses perfect knowledge of the individual's location, the individual's privacy becomes a matter for regulatory- and policy-based systems rather than obfuscation- or anonymity-based systems.

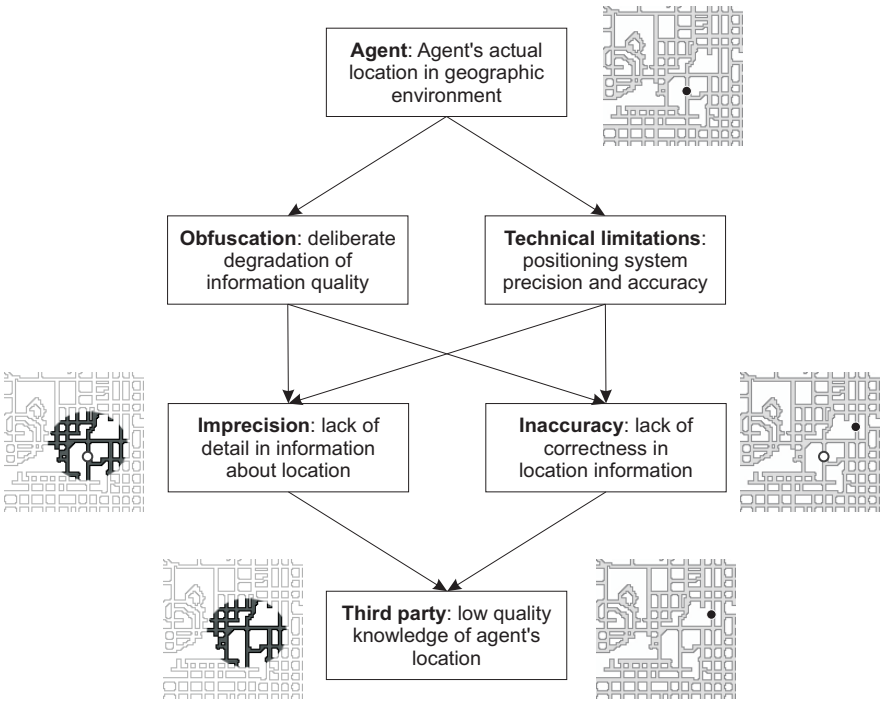

Fig. 1. Summary of location privacy scenario

In this paper we focus primarily on answering question 1. However, the answer to question 2 is also a important focus and intended future outcome of this research. Further, we argue that the answer to question 2 depends, in part, on a thorough understanding of the answer to question 1.

3.1 Refinement through assumptions

There are a variety of ways in which a 3P might attempt to gain more information about an agent's location. A 3P may have access to additional information, for example generated by secret surveillance devices or through processing of an agent's device's mobile IP address (e.g., [3]). However, in this paper we assume the only sources of information to which a 3P has access are: 1) information about the geographic environment; and 2) imperfect information about an agent's location over time. Thus, the only way for the 3P to refine its knowledge of an agent's location is by making assumptions about the spatial and temporal characteristics of that agent's movements. Some common assumptions might include:

1. *Maximum/minimum/constant speed*: A 3P might assume certain values for the agent's maximum, minimum, or average speed of movement, or assume that the agent travels at constant speed.

2. *Direction*: A 3P might assume that the agent is heading in a particular direction (e.g., north, or as near as possible to it at each intersection).
3. *Goal-directed movement*: A 3P might assume that an agent is engaged in goal-directed movement, for example, taking the shortest path from some (unknown) source to some (unknown) destination.
4. *Connected locations*: A 3P might assume that an agent can only access locations that are spatially connected (for example, accessing locations which are connected by road) or spatiotemporally connected (for example, accessing two locations connected by a railway line at a time when a train service between those locations is operating).
5. *Temporal granularity*: A 3P might assume that its knowledge about the agent's location is at a suitably fine temporal granularity such that the 3P has complete, if imperfect, knowledge of the agent's location (i.e., there are no gaps in the 3PK and the agent never manages to "sneak away" to locations outside the spatial extent of the 3PK).

While this list of assumptions is by no means complete, it gives a flavor of the diversity of potential spatial and temporal assumptions a 3P might make. Each of these assumptions may be used to refine information about an agent's location. For example, a 3P might possess imprecise information about the location of an agent at two times, t_1 and t_2. However, if the 3P assumes that an agent has a maximum speed of movement, it may be possible to discount some of the locations for time t_2 as being impossible to reach in the available time from any of the locations reported for t_1. In general, the more numerous and stronger assumptions a 3P makes, the more it can refine its information about an agent's location. At the same time, more numerous and stronger assumptions make it more likely that one or more assumptions will be invalid (e.g., an agent actually travels faster than the maximum assumed speed). Invalid assumptions will introduce inaccuracy into the 3PK.

In the following section we formalize our location privacy system. This system is able to model a variety of assumptions a 3P might make in order to refine its knowledge about an agent's location, including those presented above. The key observation is that the assumptions can be formalized into two classes, *local* and *global* assumptions, each of which have differing formal properties and computational characteristics.

4 Formal model

Section 3 introduced a number of potential assumptions that a third party might make in order to refine its knowledge about an agent's location. These assumptions are formalized in this section as *refinement operators*. The refinement operators take a 3PK as input, and refines this knowledge by excluding those possible agent locations that are inconsistent with the assumptions. The formal analysis categorizes the assumptions introduced in Section 3 into two classes: *global* and *local refinement operators*. After introducing the model foundations, we introduce the general structure and properties of refinement operators (Section 4.2). The class of global refinement operators (Section 4.3) and the class of local refinement operators (Section 4.4) are then presented along

with specific examples of how each of the assumptions in Section 3 can be formalized using a refinement operator.

4.1 Model foundations

The foundations of the formal model are as follows:

Definition 1. *A* geographic environment *is modeled as a graph $G = (V, E)$ where V is a set of vertices and $E \subseteq V \times V$ is a set of edges connecting vertices. Each edge of a graph is associated with a non-negative weight using a weighting function $w : E \rightarrow \mathbb{R}^+$, representing the distances along edges within the graph.*

Definition 2. *An* agent locator *is a function $l : T \rightarrow E$ where T is a finite set of discrete totally ordered time instants $T = \{t_1, ..., t_n\}$ and E is the set of edges in the graph G.*

Definition 3. *A* knowledge function *models a third party's knowledge of the location of an agent as the function $k_l : T \rightarrow 2^E$, where 2^E is the powerset of E.*

There are several points to note about the choice of these structures as model foundations. First, in keeping with previous work (e.g., [4,5]), the geographic environment is modeled as a graph. Graphs are a natural choice for this task in location-based services as they may be used to model constraints to movement (such as posed by a road network) and non-metric spaces (such as travel-time networks); may be embedded within two- or higher-dimensional Euclidean space; and are computationally efficient discrete structures [4,13]. In later sections, the following simple definitions drawn from standard graph theory are required.

- $ShortestPaths(G)$ denotes the set of all shortest paths (sequences of adjacent vertices $(v_1, ..., v_n)$) in the graph G. Computing the set of all shortest paths for a graph is known as the *all-pairs shortest path* (APSP) problem. A number of common $O(n^3)$ algorithms exist for computing APSPs, including the Bellman-Ford algorithm, Floyd's algorithm, or simply iterating Dijkstra's algorithm [2].
- $IsReachable(V, G)$ denotes the set of vertices in the graph G that can be reached by some path in G from at least one vertex in V within a single "clock tick." In defining $IsReachable$, we therefore assume that the time domain T has a minimum granularity (akin to the concept of "chronons" in spatial databases). Similarly, $IsReachable(E, G)$ denotes the set of edges in the graph G that can be reached by some path in G from at least one edge in E within a single clock tick. Reachable vertices or edges can be found using standard shortest path algorithms (possibly requiring additional assumptions about the speed of travel).
- $Connected(V, G)$ denotes the set of vertices in the graph G that are connected by some path in G to at least one vertex in V. Similarly, $Connected(E, G)$ denotes the set of edges in G that are connected by some path in G to at least one edge in E. Again, standard algorithms exist for finding the set of connected locations in a graph, including computing the transitive closure of the graph.
- $Incident(e, p)$ indicates that an edge e is incident with path p (i.e., e has at least one vertex in common with p). Similarly, the symbol $Incident(E, p)$ indicates that the edge set E contains at least one edge which is incident with path p.

Second, the agent locator represents the location of an agent at a particular time instant as an *edge* of the graph (rather than the more obvious choice of a node). Edges are preferred as locations in our formal model because this approach:

- provides an inherently qualitative representation of location that does not assume the existence of precise coordinates (useful, for example, where location-sensing technology like GPS cannot provide completely precise and accurate location information, even if perhaps augmented with map matching techniques);
- can be extended to represent continuous movement along an edge, rather than assuming instantaneous movement between adjacent nodes (acknowledged as a drawback in previous work, [4]); and
- leads to an efficient and compact formal structure.

At the same time, by subdividing edges with additional vertices (termed *graph homomorphisms*), arbitrarily fine spatial detail can still be represented by the edges in a graph.

Finally, a knowledge function represents a third party's knowledge of the location of an agent as a set of edges for a particular time t. The set of all times T is discrete and finite. These times provide the smallest granularity of time that may be represented in the system (similar to the concept of a *chronon* or *tick* in spatiotemporal databases). The set of knowledge functions has a number of characteristic features, including:

- A knowledge function k_l is *accurate* if $l(t) \in k_l(t)$ for all $t \in T$; otherwise the knowledge function is *inaccurate*.
- A knowledge function k_l is *precise* if $|k_l(t)| = 1$ for all $t \in T$; otherwise the knowledge function is *imprecise*.
- A knowledge function k_l contains *ignorance* if there exists a t such that $k_l(t) = E$, where E is the set of all edges. Ignorance is a special case of imprecision.
- A knowledge function k_l is *inconsistent* if there exists a t such that $k_l(t) = \varnothing$; otherwise a knowledge function is *consistent*.

For example, Figure 2 depicts a simple graph (geographic environment) $G = (V, E)$ with 25 edges (locations). Note that a geographic environment need not be connected. The movement of an agent for time domain $T = \{1, 2, 3\}$ from edge 2 at time 1 along edge 8 at time 2 to edge 17 at time 3 is represented by the function $l = \{(1, 2), (2, 8), (3, 17)\}$. A 3PK of the agent's movement might be represented by the function $k_l = \{(1, \{2, 3\}), (2, E), (3, \{17, 8, 16\})\}$. At any particular time the third party does not know exactly on which edge the agent is located: the agent has a degree of location privacy. In this case, the knowledge function k_l is imprecise, accurate, consistent, and contains ignorance.

4.2 Refinement operators

In this section we provide some preliminary definitions leading up to the general definition of a refinement operator. First, we define the knowledge function space as the set of all knowledge functions.

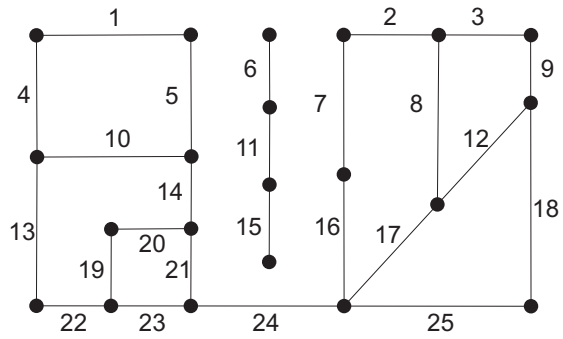

Fig. 2. Example geographic environment graph

Definition 4. *The* knowledge function space K *is the set of all knowledge functions* $(2^E)^T$*, i.e.,* $K = \{k | k : T \rightarrow 2^E\}$.

Next we define a relation $\leq$ on the set of knowledge functions K:

Definition 5. *The relation* $\leq$ *is a relation on the set of knowledge functions* K *such that* $k_1 \leq k_2$ *if and only if* $k_1(t) \subseteq k_2(t)$ *for all* $t \in T$.

It is easy to see that the relation $\leq$ forms a partial order, since $\leq$ is reflexive ($k \leq k$), antisymmetric ($k_1 \leq k_2 \wedge k_2 \leq k_1 \rightarrow k_1 = k_2$), and transitive ($k_1 \leq k_2 \wedge k_2 \leq k_3 \rightarrow k_1 \leq k_3$). Finally, we define a refinement operator as follows:

Definition 6. *A* refinement operator $\mathcal{R}$ *is a function* $\mathcal{R} : K \rightarrow K$ *such that* $\mathcal{R}(k) \leq k$ *and* $\mathcal{R}(k) \leq \mathcal{R}(\mathcal{R}(k))$. *A refinement operator is said to be* monotonic *if* $k_1 \leq k_2 \rightarrow \mathcal{R}(k_1) \leq \mathcal{R}(k_2)$.

The property that $\mathcal{R}(k) \leq k$ is a defining property of refinement: applying an assumption should always increase or maintain the precision of a 3PK, but never decrease its precision. The property $\mathcal{R}(k) \leq \mathcal{R}(\mathcal{R}(k))$ provides idempotency: refinement operators must not increase the precision of a 3PK through repeated application. Monotonicity is a useful property that is exhibited by all the refinement operators discussed in this paper. A monotonic operator is always idempotent. However, in general it is possible to conceive of non-monotonic refinement operators. Based on this general definition of refinement, in the following sections we provide definitions for two distinct classes of refinement operators: global and local refinement.

4.3 Global refinement

A *global refinement operator* $\Gamma : K \rightarrow K$ is a refinement operator that uses the entirety of a third party's knowledge about an agent's location over time in order to compute a refinement of the agent's location over time. The key feature of a global refinement, when compared with local refinement, is that it utilizes the 3PK in a *single step*. There are potentially any number of different global refinement operators that might be defined based on this general specification. In the context of location privacy, we discuss two natural global refinements: connected refinement and goal-directed refinement.

Connected refinement A basic assumption introduced in Section 3 is that an agent can only access locations that are connected in the geographic environment. Under this assumption an agent can never be located at any edge that is disconnected from any other edge at which it has been located. This concept can be formalized as a global refinement operator $\Gamma_C : K \rightarrow K$ as follows:

$$\Gamma_C(k_l)(t) = \Big(\big(\bigcap_{t' \in T} Connected(k_l(t'), G) \big) \cap k_l(t) \Big)$$

The operator Γ_C is well formed in that it satisfies the basic properties of refinement operators, i.e., $\Gamma_C(k) \leq k$ and $\Gamma_C(k) \leq \Gamma_C(\Gamma_C(k))$. Further, for any sets of edges E_1, E_2 of a graph $G = (V, E)$ then $E_1 \leq E_2 \rightarrow Connected(E_1, G) \subseteq Connected(E_2, G)$. It follows that Γ_C is a *monotonic* refinement operator.

To illustrate, consider again Figure 2. Given the knowledge function $k_l = \{(1, \{1, 5, 6\}), (2, \{10, 5, 14\}), (3, \{20, 21, 15\})\}$, and assuming the agent can only travel between connected edges in the graph, then the agent cannot be located at edges 6 or 15. Formally: $\Gamma_C(k_l) = \{(1, \{1, 5\}), (2, \{10, 5, 14\}), (3, \{20, 21\})\}$.

Goal-directed refinement Another assumption proposed in Section 3 is that an agent is moving in a goal-directed manner, e.g., it is taking the shortest path between two (unknown) locations. Under this assumption, the agent's true location can only be amongst those locations that are incident with at least one shortest path that travels through all previous known location sets. This assumption can be formalized as a global refinement operator $\Gamma_D : K \rightarrow K$ as below.

$$\Gamma_D(k_l)(t) = k_l(t) \cap \{e \in E | \exists p \in ShortestPaths(G). \\ \big(\forall t' \in T. Incident(e, p) \wedge Incident(k_l(t'), p)\big)\}$$

By definition Γ_D satisfies the defining properties of a refinement operator. The operator Γ_D is also monotonic as a consequence of the basic properties of the incidence relation.

To illustrate, consider $k_l = \{(1, \{3, 9\}), (2, \{12, 17\}), (3, \{7, 16\})\}$. Assuming Euclidean distances as edge weights for the graph in Figure 2 then $\Gamma_D(k_l) = \{(1, \{3, 9\}), (2, \{12, 17\}), (3, \{16\})\}$. There is no shortest path from edge 9 or 3, passing through edge 12 or 17 which also passes through edge 7.

4.4 Local refinement

Operators belonging to the local refinement class are built up from the recursive application of atomic refinement operations. Unlike global refinement operators, local refinement operators construct a refined knowledge function piecewise, considering consecutive time instants. The most basic component of a local refinement operator is the *alpha combination* of two sets of edges at consecutive time instants. Based on the knowledge of an agent's location at t_{i-1} and t_i, the alpha combination provides the refinement of the agent's location at time t_i. Specific local refinement operators work by providing

specific definitions for the alpha-combinations. The next level of the local refinement operator is the *beta combination* of a knowledge function between two time instants. Beta combinations recursively combine alpha combinations. Formally:

Definition 7. *An* alpha-combination *is a function* $\alpha : 2^E \times 2^E \rightarrow 2^E$.

In order to be well formed, the alpha combination must have the properties that $\alpha(E_1, E_2) \subseteq E_2$ (refinement property) and $E_1' \subseteq E_1 \rightarrow \alpha(E_1', E_2) \subseteq \alpha(E_1, E_2)$ (monotonic with respect to first argument). Alpha-combinations of (local) pairs of edge sets are recursively combined using *beta-combinations*, defined as follows:

Definition 8. *A* beta-combination *is a function* $\beta : K \times T \times T \rightarrow K$*, where*

$$\beta : (k, t_s, t_j) \mapsto \begin{cases} \textit{if } j = s & \big\{\big(t_s, k(t_s)\big)\big\} \\ \textit{if } j = s+1 & \beta(k, t_s, t_s) \cup \big\{\big(t_{s+1}, \alpha(k(t_s), k(t_{s+1}))\big)\big\} \\ \textit{otherwise} & \beta(k, t_s, t_{j-1}) \cup \\ & \big\{\big(t_j, \alpha(\beta(k, t_s, t_{j-1})(t_{j-1}), k(t_j))\big)\big\} \end{cases}$$

Finally, the top level local refinement operator returns a beta-combination:

Definition 9. *For any* $T = \{t_0, ..., t_n\}$ *a* local refinement operator Λ *is a function*

$$\Lambda : K \rightarrow K,\ k_l \mapsto \beta(k_l, t_0, t_n)$$

Temporal granularity refinement An assumption introduced in Section 3 is that the 3PK is at a temporal granularity fine enough to preclude the possibility that the agent has ever traveled outside the spatial extent of the 3PK. This assumption may be modeled as a local refinement operator Λ_G where for each pair of consecutive times t_i, t_{i+1} the agent can only be located at those edges that are connected in the subgraph formed by the union of $k_l(t_i)$ and $k_l(t_{i+1})$. To define this operator on a graph $G = (V, E)$ simply requires a redefinition of the alpha combination used for Λ_G as follows:

$$\alpha_G(E_1, E_2) = \mathit{Connected}(E_1, (V, E_1 \cup E_2))$$

Because $\alpha_G(E_1, E_2) \subseteq E_2$, Λ_G satisfies the defining properties of a refinement operator, and is monotonic in a similar way to the global connected refinement operator.

For the knowledge function $k_l = \{(1, \{1, 5\}), (2, \{13, 14\}), (3, \{16, 21\})\}$ (see Figure 2) the following the series of steps leads to the temporal granularity refinement $\Lambda_G(k_l) = \{(1, \{1, 5\}), (2, \{14\}), (3, \{21\})\}$:

$$\begin{aligned} \Lambda_G(k_l) &= \beta(k_l, 1, 3) \\ &= \beta(k_l, 1, 2) \cup \{\big(3, \alpha_G(\beta(k_l, 1, 2)(2), k_l(3))\big)\} \\ &= \{(1, \{1, 5\}), (2, \{14\})\} \cup \{(3, \alpha_G(\{14\}, \{16, 21\}))\} \\ &= \{(1, \{1, 5\}), (2, \{14\}), (3, \{21\})\} \end{aligned}$$

Maximum speed refinement Given $T \subset \mathbb{R}$ and a weighting function w associated with the graph G which represents the distance along each edge, the travel time along each edge for an agent moving at a specified maximum speed can easily be calculated in order to decide whether two edges are reachable from one another. The assumption that an agent can only travel at a specified maximum speed (Section 3) forms the basis of the maximum speed refinement operator Λ_S. Again, Λ_S depends on the definition of the alpha combination for the operator, as follows:

$$\alpha_S(E_1, E_2) = IsReachable(E_1, G) \cap E_2$$

For similar reasons to the global goal-directed refinement operator, Λ_S satisfies the defining properties of a refinement operator and is monotonic. Definitions for further local refinement operators, including those derived from minimum speed and direction assumptions, are omitted here, but may be constructed in a similar way.

4.5 Combining operators

One of the goals of providing definitions for refinement operators is to enable the combination of different refinement operators in a structured and predictable way. Consider the set of all refinements (global or local) of a knowledge function k. A pair of refinements $\mathcal{R}_1$ and $\mathcal{R}_2$ of k may be combined by taking the intersection of the two independent refinements. This refinement combination $\circ$ can be defined as follows:

Definition 10. *For two refinement operators $\mathcal{R}_1$ and $\mathcal{R}_2$ and a knowledge function k, let $\mathcal{R}_1(k)$ and $\mathcal{R}_2(k)$ be the corresponding refined knowledge functions. The combination operator $\circ$ of two knowledge functions $\mathcal{R}_1(k)$ and $\mathcal{R}_2(k)$ is then defined as*

$$(\mathcal{R}_1(k) \circ \mathcal{R}_2(k))(t) = (\mathcal{R}_1(k))(t) \cap (\mathcal{R}_2(k))(t).$$

As a consequence of the properties of set intersection, the operator $\circ$ is commutative and associative for all refinement functions of a knowledge function k. Further, consider the empty refinement $\mathcal{E}(k)$, where $\mathcal{E}$ is a local refinement operator with alpha combination $\alpha_{\mathcal{E}}(E_1, E_2) = E_2$. This refinement forms an *identity* for the combination $\circ$, i.e., $\mathcal{R}(k) \circ \mathcal{E}(k) = \mathcal{R}(k)$. Consequently, $\circ$ has the algebraic structure of a *monoid* (closure, commutativity, associativity, and identity) on the set of all refinements of a knowledge function k.

In addition, local refinement operators may be combined by taking the intersection of alpha combinations from each of the two refinement operators, resulting in the combination $\bullet$ as follows:

Definition 11. *For two local refinement operators Λ_1 and Λ_2 with alpha combinations α_1 and α_2, respectively, and a knowledge function k, $\Lambda_1(k) \bullet \Lambda_2(k)$ is the local refinement operator with alpha combination $\alpha_\bullet \equiv \alpha_1 \cap \alpha_2$.*

Again, we can show that $\bullet$ forms a monoid, with $\mathcal{E}(k)$ as the identity element.

Lemma 1. *For two local refinement operators Λ_1, Λ_2 and a knowledge function k, then:*

$$\Lambda_1(k) \bullet \Lambda_2(k) \leq \Lambda_1(k) \circ \Lambda_2(k)$$

Proof. By induction. Let $T = \{t_0, \ldots, t_n\}$, $i \in \{0, \ldots, n\}$. Let α_1, β_1 and α_2, β_2 be the alpha and beta combinations of Λ_1 and Λ_2, respectively. Let $\beta_{\alpha_1 \cap \alpha_2}$ denote the beta combination of $\alpha_1 \cap \alpha_2$. The lemma is trivially true for $i = 0$, because all beta combinations depend in this case on k only. If $i = 1$, then

$$\beta_{\alpha_1 \cap \alpha_2}(k, t_0, t_1)(t_1) = \beta_1(k, t_0, t_1)(t_1) \cap \beta_2(k, t_0, t_1)(t_1).$$

If the lemma is true for $i - 1 > 1$, i.e.

$$\beta_{\alpha_1 \cap \alpha_2}(k, t_0, t_{i-1})(t_{i-1}) \subseteq \beta_j(k, t_0, t_{i-1})(t_{i-1}), \qquad j = 1, 2,$$

then we obtain

$$\begin{aligned} &\alpha_1\big(\beta\alpha_1 \cap \alpha_2(k, t_0, t_{i-1})(t_{i-1}), k(t_i)\big) \\ \cap\, &\alpha_2\big(\beta\alpha_1 \cap \alpha_2(k, t_0, t_{i-1})(t_{i-1}), k(t_i)\big) \\ &\subseteq \alpha_j\big(\beta_j(k, t_0, t_{i-1})(t_{i-1}), k(t_i)\big) \qquad j = 1, 2, \end{aligned}$$

because alpha combinations are monotonic with respect to their first argument. This implies

$$\beta_{\alpha_1 \cap \alpha_2}(k, t_0, t_i)(t_i) \subseteq \beta_j(k, t_0, t_i)(t_i), \qquad j = 1, 2,$$

and therefore

$$\beta_{\alpha_1 \cap \alpha_2}(k, t_0, t_i)(t_i) \subseteq \beta_1(k, t_0, t_i)(t_i) \cap \beta_2(k, t_0, t_i)(t_i),$$

which proves the lemma.

The importance of the algebraic properties of combinations of refinement operators is that for a given knowledge function we can say that the order in which refinement operators are applied will not affect the results of the refinement itself.

5 Computational model

The formal model presented above has intentionally been kept simple. As such, several aspects of the formal model are not computationally viable. Here we discuss some of the steps taken to implement privacy simulation software based on the formal model. The aim of the simulation environment is to show how the formal model can be translated into a practical computational model, and to allow empirical exploration of different privacy protection and invasion strategies in future work. First we will present some general observations concerning the development of the simulation environment. Then we discuss the implementation of global operators, followed by the implementation of local operators. Finally, we examine some methods for extending the formal model to accommodate more complicated assumptions about the behavior of an agent.

5.1 The simulation environment

Our simulation environment is a simple model of a mobile phone network. Streets are represented by the edges of a graph, along which an agent moves. The location of an agent is not precisely known but is given by the set of all streets that partially lie within a cell of the phone network. The simulation environment enables the application of a variety of refinements to a 3PK in order to generate a more precise and accurate representation of the agent's current location.

The simulation is based on discrete events. At each time instant (tick) the simulation determines the set of edges that overlap with the cell in which the agent is currently located. This edge set forms the 3PK, called the knowledge base (KB) entry, for that time instant. The KB is implemented as a list of sets of edges. The list structure performs the role of the time values in the formal model—ordering the edge sets. Additional information, such as time deltas, could easily be added to the KB if required.

In our simulation, the KB is updated at each tick with new knowledge about the agent's location. The refinement operators are applied after each update. To increase computational efficiency, the KB is only updated when the set of potential agent locations has changed for many refinement operators. This corresponds to the point in time when the agent is detected as having transitioned between two cells. By minimizing KB entries in this manner, we minimized both the number of calls to our refinement functions and the size of the KB when those refinement functions were invoked. Some refinement operators, for example the "maximum speed" operator cannot be optimized this way as they may require the KB to be updated even if the set of potential edges remains unchanged. The implementation of the "maximum speed" operator is discussed at the end of this section.

5.2 Global operators

Two global operators were implemented for this simulation: the "goal directed" operator and the "connected path" operator. These operators are defined in Section 4.3. One optimization in our implementation is to precompute all pairs of shortest path in the graph. Used this precomputed result, Algorithm 1 returns all edges in the graph that are connected to at least one of the elements in an input set of edges.

On the basis of Algorithm 1, we can define the algorithm for computing the connected path refinement operator, *ConnectedPath* (Algorithm 2). The first loop in Algorithm 2 builds the set S of all edges in the graph that are connected to at least one edge in the input KB. The second loop intersects S with each edge set in the input KB to generate the refined KB.

The directed path refinement operator is computed by Algorithm 3. The first loop finds all the shortest paths that share at least one edge with every set of edges in our input KB. The edges of each path that satisfies this criterion are added to an edge set S. In the second loop, the algorithm simply intersects each unrefined edge set with S to produce a refined edge set.

Algorithm 1: Connected algorithm, *Connected*(S)

Data: S, a set of edges
Result: R, a set of edges connected to at least one edge in S
Local variables: a list of shortest paths L; shortest path s; nodes n, n_1, n_2; edges e, e'
$R \leftarrow \varnothing$;
for *each edge* $e \in S$, *where* $e = (n_1, n_2)$ **do**
  $L \leftarrow$ *GetAllShortestPathsFrom*$(n_1) \cup$ *GetAllShortestPathsFrom*(n_2);
  for *each* $s \in L$ **do**
    $n \leftarrow$ *GetNodeAtEndOfPath*(s);
    for *each edge* e' *starting from* n **do**
      $R \leftarrow \{e'\} \cup R$;

Algorithm 2: Connected path algorithm, *ConnectedPath*(K_p)

Data: K_p, the knowledge base (ordered list of sets of edges) to process
Result: K_o, the refined knowledge base
Local variables: sets of edges S, S'
$S \leftarrow \varnothing$;
$K_o \leftarrow K_p$;
for *each edge set* $S' \in K_o$ **do**
  $S \leftarrow S \cup$ *Connected*(S');
for *each edge set* $S' \in K_o$ **do**
  $S' \leftarrow S' \cap S$;

5.3 Local operators

A key feature of local operators is their recursive nature. Implementations of the local operators can take advantage of this recursion to improve operator performance, by only processing updates to the knowledge function each time the operator is called. Algorithm 4 implements the generic local operator. The algorithm requires one argument, a KB K_l that is persistently stored externally to the algorithm. Whenever new edge sets are added to the unrefined KB, we also add the same edge sets to our stored KB K_l. Thus, for the first call of the algorithm, K_l will simply be a copy of the unrefined KB. The global index l is initially set to 1, so the entire KB will be processed. The final line of the Algorithm 4 stores the cardinality of the KB K_l in the global index l. So, for subsequent calls of the algorithm, only new, unrefined edge sets are processed, increasing the efficiency of the local operator implementation.

In Section 4.5 we defined the combination of local operators using the $\bullet$ operator. A combination of operators can be implemented using a single refined KB, shared between all local operators. The result of each operator's refinement is intersected with the shared KB at every time step. In this manner, processing a local operator still only requires an operation on a single pair of edge sets. However, if the active local operators change, the shared KB needs to be rebuilt from the start.

Algorithm 3: Direct path algorithm, *DirectedPath*(K_p)

Data: K_p, the knowledge base (ordered list of sets of edges) to process
Result: K_o, the refined knowledge base
Local variables: sets of edges S, T, U, V; shortest path s; Boolean b
$V \leftarrow \varnothing$;
$K_o \leftarrow K_p$;
for *each shortest path* s *in the graph* **do**
  $b \leftarrow$ *true*;
  $T \leftarrow$ *GetPathEdges*(s);
  for *each edge set* $U \in K_p$ **do**
    if $(|T \cap U| = 0)$ **then** $b \leftarrow$ *false*;
  if b **then** $V \leftarrow V \cup T$;
for *each edge list* $S \in K_o$ **do**
  $S \leftarrow S \cap V$;

Algorithm 4: Local processing algorithm, *LocalRef*(K_l)

Data: K_l, externally stored knowledge base to process (possibly partially refined)
Result: K_l, the fully refined version of the input knowledge base
Static variables: global index l initialized as $l \leftarrow 1$
Local variables: local index i
for $(i \leftarrow l;\ i < |K_l|;\ i = i + 1)$ **do**
  $K_l[i] \leftarrow$ *DoAlphaOperation*$(K_l[i-1], K_l[i])$;
$l \leftarrow |K_l|$;

5.4 Extending the base model

The computational model outlined above can be extended to handle more complicated assumptions. As an example, we discuss the implementation of the maximum speed local refinement operator. Formally, this operator is defined by the set of edges that are reachable from any edge of the KB entry in a single tick, assuming some maximum speed. As mentioned earlier, new entries in the KB are only added when the agent transitions between cells. However, edges may become reachable over time even if the unrefined KB does not change. The maximum speed operator therefore needs to refine the KB (i.e., perform an alpha operation) at each tick.

Two specific problems arose in implementing this extension. First, it is unlikely that the time between each tick will be uniform. To achieve framerate-independent refinement, the simulation must take into account the actual time delta between ticks. Second, as an agent can only be located to the precision of an edge, its precise location on the edge is unknown. A naive implementation then allows an agent can reach all adjacent edges in zero time, with the unfortunate side-effect that the refined set of potential agent edges will expand recursively at each update frame, no matter how small the time delta or how slow the agent is (see Figure 3).

The first problem was solved by determining the set of reachable edges based on physical speeds (meters per second) rather than tick speed (meters per tick). This implied a modification of the alpha operator implementation to accept a time delta in

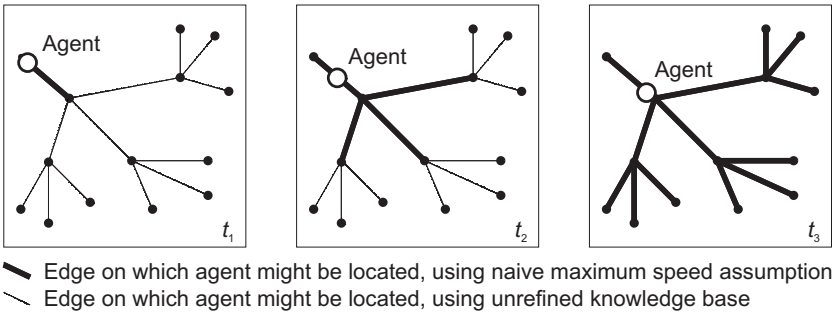

Fig. 3. A naive implementation of the maximum speed operator expands the potential agent location set at each tick, regardless of any assumed maximum speed

Tick	Tot. time	Time Δ	Edge Set	Processing
1	0	11	E_1	—
2	11	10	E_2	$a(E_1, E_2, (11-0)+10)$
3	21	9	E_2	$a(E_1, E_2, (11-0)+10+9)$
4	30	9	E_3	$a(R(3), E_3, (30-11)+9)$
5	39	11	E_4	$a(R(4), E_4, (39-30)+11)$
6	40	10	E_4	$a(R(4), E_4, (39-30)+11+10)$
7	50	12	E_4	$a(R(4), E_4, (39-30)+11+10+12)$
8	62	10	E_5	$a(R(7), E_5, (62-39)+10)$

Fig. 4. Example update calls for the maximum speed operator. Function a, a modified implementation of an α-combination for a maximum speed local refinement operators, accepts two edge sets and a time delta; R returns the refined knowledge base for a particular tick.

addition to the two edge sets. The second problem was solved by simply discarding our refinements until the KB actually changed. This meant that we would always update with respect to the "original" refinement, albeit with a constantly increasing time delta. A simple example is provided in Figure 4.

6 Discussion

This paper has set out a formal model of location privacy based on obfuscation. This model extends previous work in two ways:

1. The approach models both the strategies an individual can employ to protect his or her location privacy and the counter strategies that a 3P might employ to subvert these strategies. Counter strategies require that the 3P make one or more assumptions about the movement of the agent. These assumptions and their combinations are formally modeled as refinement operators. Combinations of refinement operators have predictable algebraic properties, ensuring that the order in which operators are combined does not affect the results of that combination.
2. The approach adopts an explicitly spatiotemporal model of location privacy. A 3P can use knowledge of an individual's movement over time to build refinements

of that individual's location. Two distinct classes of refinement operator, global and local, are identified. These refinement operator classes differ in the way they handle spatiotemporal information. Global refinement operators work on the entire history of movement in a single step. Local refinement operators work recursively on knowledge of changing location, one clock tick at at time.

The paper has also indicated how some of the over-simplifications inherent in the formal model can be overcome when the formal model is translated into a computational model, implemented as a simulation environment.

Future work will focus on the empirical evaluation of location privacy protection and invasion strategies by experimenting using simulations of various 3P assumptions upon mobile agents. These experiments will investigate the trade-off between accuracy and precision of 3PK. Increased use of assumptions by a 3P will increase the precision of 3PK about the location of an agent. However, using incorrect assumptions may introduce inaccuracies into the 3PK. Thus, achieving a balance between the most precise but still accurate refined knowledge of an individual's location is the key to effective invasion of location privacy (and so understanding this balance is the key to effective counter-strategies to invasion of location privacy).

Further work will also investigate other strategies an individual might use to counter the refinement strategies of a 3P. For example, certain routes through a geographic environment might be more private than others (e.g., ones that are better connected might provide better protection from temporal granularity refinement operators). Understanding how refinement strategies operate is a prerequisite to understanding how an individual might protect against these refinements. One final open question that has not been addressed in this paper is the interaction between mobile individuals. Individuals do not move in isolation of other individuals, rather our movements are to some extent coordinated by spatiotemporal constraints, such as rush hour traffic or attendance at a soccer match or large concert. Where a 3P has (imperfect) knowledge of the location of many individuals, the 3P may be able to refine its knowledge based on inferences between individuals within groups. Modeling the aggregate movement of changing groups of individuals is an important future challenge.

Acknowledgments

Dr Duckham is partially supported under the Australian Research Council's Discovery funding scheme (project number DP0662906). Dr Kulik is partially supported by an Early Career Researcher Grant from the University of Melbourne.

References

1. A.R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
2. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.
3. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation Onion router. In *Proc. 13th USENIX Security Symposium*, 2004.

4. M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In H.W. Gellersen, R. Want, and A. Schmidt, editors, *Pervasive 2005*, volume 3468 of *Lecture Notes in Computer Science*, pages 152–170. Springer, Berlin, 2005.
5. M. Duckham and L. Kulik. Simulation of obfuscation and negotiation for location privacy. In D.M. Mark and A.G. Cohn, editors, *COSIT 2005*, volume 3693 of *Lecture Notes in Computer Science*, pages 31–48. Springer, Berlin, 2005.
6. M. Duckham and L. Kulik. Location privacy and location-aware computing. In J. Drummond, R. Billen, D. Forrest, and E. João, editors, *Dynamic and Mobile GIS: Investigating Change in Space and Time*, chapter 3. CRC Press, Boca Raton, FL, 2006.
7. S. Duri, M. Gruteser, X. Liu, P. Moskowitz, R. Perez, M. Singh, and J-M. Tang. Framework for security and privacy in automotive telematics. In *Proc. 2nd International Workshop on Mobile Commerce*, pages 25–32. ACM Press, 2002.
8. F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore, and M. Bylund. GeoNotes: Social and navigational aspects of location-based information systems. In G.D. Abowd, B. Brumitt, and S. Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 2001.
9. W. W. Görlach, A. Terpstra and A. Heinemann. Survey on location privacy in pervasive computing. In *Proc. First Workshop on Security and Privacy at the Conference on Pervasive Computing (SPPC)*, 2004.
10. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. MobiSys '03*, pages 31–42, 2003.
11. M. Gruteser and D. Grunwald. A methodological assessment of location privacy risks in wireless hotspot networks. In D. Hutter, G. Müller, and W. Stephan, editors, *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 10–24. Springer, 2004.
12. J. I. Hong and J. A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *Proc. 2nd International Conference on Mobile Systems, Applications, and Services*, pages 177–189. ACM Press, 2004.
13. C. S. Jensen. Database aspects of location-based services. In J. Schiller and A. Voisard, editors, *Location-based services*, chapter 5, pages 27–39. Morgan Kaufmann, 2004.
14. E. Kaasinen. User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7(1):70–79, 2003.
15. M. Langheinrich. Privacy by design—principles of privacy-aware ubiquitous systems. In G. D. Abowd, B. Brumitt, and S. Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 273–291. Springer, 2001.
16. M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In G. Borriello and L. E. Holmquist, editors, *UbiComp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, pages 237–245. Springer, 2002.
17. R.R. Muntz, T. Barclay, J. Dozier, C. Faloutsos, A.M. Maceachren, J.L. Martin, C.M. Pancake, and M Satyanarayanan. *IT Roadmap to a Geospatial Future*. The National Academies Press, Washington, DC, 2003.
18. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2001.
19. B. N. Schilit, J.I. Hong, and M. Gruteser. Wireless location privacy protection. *IEEE Computer*, 36(12):135–137, 2003.
20. E. Snekkenes. Concepts for personal location privacy policies. In *Proc. 3rd ACM conference on Electronic Commerce*, pages 48–57. ACM Press, 2001.
21. A. F. Westin. *Privacy and freedom*. Atheneum, New York, 1967.