On the Transit Obfuscation Problem

Hideaki Takahashi
The University of Tokyo
Tokyo, Japan
takahashi-hideaki567@g.ecc.u-tokyo.ac.jp

Alex Fukunaga
The University of Tokyo
Tokyo, Japan
fukunaga@idea.c.u-tokyo.ac.jp

ABSTRACT

Concealing an intermediate point on a route or visible from a route is an important goal in some transportation and surveillance scenarios. This paper studies the Transit Obfuscation Problem, the problem of traveling from some start location to an end location while "covering" a specific transit point that needs to be concealed from adversaries. We propose the notion of transit anonymity, a quantitative guarantee of the anonymity of a specific transit point, even with a powerful adversary with full knowledge of the path planning algorithm. We propose and evaluate planning/search algorithms that satisfy this anonymity criterion.

KEYWORDS

obfuscation, deceptive planning, planning

ACM Reference Format:
Hideaki Takahashi and Alex Fukunaga. 2024. On the Transit Obfuscation Problem. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

In applications such as sensitive cargo transportation or surveillance, it is sometimes necessary to route an agent from a start point to a goal while concealing the location of a *transit point*, which is either on the route or visible from the route, from adversaries. For example, in a cargo transport application, if a depot or drop location is located somewhere on the route, it is essential to prevent potential adversaries from deducing the transit point location to minimize the risk of theft or interception. In a surveillance application, it is important to be able to conceal which specific location a surveillance agent is targeting on its route.

Obfuscating an agent's true intention has been previously studied in various fields, including path planning, robotics, and game theory [2]. Previous work has primarily focused on the Goal Obfuscation Problem, which aims to prevent observers from deducing the agent's actual goal, and has numerous applications, e.g., creating realistic non-player characters (NPCs) capable of deceiving humans [4], or secure escorting of a VIP to a hidden location [6].

Concealing non-goal locations is also important for customer privacy protection. For example, Enayati et al. [5] considers a scenario where a UAV delivers packages to a private location and returns to the starting point. Chen et al. [3] notes that some public transportation systems track the stations where each customer boards and leaves, potentially revealing the location of homes or workplaces, which are the transit points on the round-trip paths. However, Enayati et al. [5] is limited to path planning in two-dimensional coordinates, and Chen et al. [3] focuses on not path planning but anonymizing the collected sequential data.

In this paper, we study the **Transit Obfuscation Problem**, where given a graph, start location, end location, a target transit point, and a visibility function for an agent, the task is to generate a route from the start to the end location such that the agent's visibility function covers the target, but the target location is concealed from adversaries. We assume a strong adversary that has full knowledge of the agent's path, as well as full knowledge of the domain as well the internal decision-making process of the agent (i.e., the adversary has full access to the agent's code).

We introduce the notion of (k, ℓ, m)-Anonymity, which quantifies the level of concealment achieved by a path planner. If a path planner satisfies (k, ℓ, m)-Anonymity, there exist at least k transit points resulting in the same path up to the first m steps, while the deviation among these candidate points is $\geq \ell$. Thus, (k, ℓ, ∞)-Anonymity is a guarantee that even with full knowledge of the agent's path and code, an adversary can not distinguish the true target transit point among k possible candidates which are at least ℓ apart from each other. We analyze some theoretical properties of (k, ℓ, m)-Anonymity and propose a *graph partitioning-based* approach to generating paths that guarantee (k, ℓ, m)-Anonymity.

The rest of the paper is structured as follows. First, we define the Transit Obfuscation Problem (TOP) with respect to a path-planning problem with a transit point and visibility constraints (Section 2) Next, in Section 3, we define (k, ℓ, m)-Anonymity for the TOP and analyze its theoretical properties. We also define some metrics for evaluating the tradeoffs between privacy and path costs for the TOP. Then, in Section 4, we propose a partitioning-based search algorithm for the TOP which guarantees anonymity even when when the adversary knows the complete path, i.e., (k, ℓ, ∞)-Anonymity. Section 5 proposes algorithms which guarantees anonymity for up to $m < \infty$ steps. In Section 6, we experimentally evaluate our search algorithms on some standard benchmark game map instances. Section 7 discusses related work. Section 8 concludes with a discussion and directions for future work. Our code is available at: https://github.com/Koukyosyumei/TOP.

2 TRANSIT OBFUSCATION PROBLEM

A **path-planning domain with visibility constraints** is denoted by a triple $\mathcal{D} = \langle \mathcal{N}, \mathcal{E}, \mathcal{T}, c, v \rangle$, where

- $\mathcal{N}$ is a non-empty set of nodes;
- $\mathcal{T} \subseteq \mathcal{N}$ is a set of transit candidates;
- $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of edges between nodes;

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). ...$ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

- $c : \mathcal{E} \rightarrow R_0^+$ is a function that returns the non-negative cost of an edge between two nodes.
- $v : \mathcal{N} \rightarrow 2^{\mathcal{N}}$ is a visibility function that returns the set of visible nodes from a given node.

The cost of the shortest path (a.k.a minimum cost path) between node $a \in \mathcal{N}$ and node $b \in \mathcal{N}$ is denoted by $d(a, b)$. For simplicity, we assume that E does not contain self-loop edges, i.e., $d(a, b) = \infty$ if $a = b$. A **path** π in a path-planning domain $\mathcal{D}$ is a sequence of nodes $\pi = n_1, n_2, ..., n_{|\pi|}$ such that $\forall i \in \{1, ..., |\pi| - 1\}$ $(n_i, n_{i+1}) \in \mathcal{E}$, where $|\pi|$ represents the length of π. We also denote the subsequence of π till m-th node as $\pi|_m$, i.e., $\pi|_m = n_1, n_2, ..., n_m$. For convenience, we assume that $\pi|_m = \pi$ if $m > |\pi|$. The binary operator $\circ$ represents the concatenation of two paths. Specifically, when $\pi_a = a_1, a_2, ..., a_{|\pi_a|}$, $\pi_b = b_1, b_2, ..., b_{|\pi_b|}$, and $a_{|\pi_a|} = b_1$, we have that $\pi = \pi_a \circ \pi_b = a_1, a_2, ..., a_{|\pi_a|}, b_2, ..., b_{|\pi_b|}$. We also introduce $\|$ notation, where $\|_{i=1}^{x} \pi_i = \pi_1 \circ \pi_2 \circ ... \circ \pi_x$. The cost of π is the sum of the costs of each edge in π, given by $cost(\pi) = \sum_{i=2}^{|\pi|} c(\pi_{i-1}, \pi_i)$, where π_i is the i-th node in π. We say that π *covers* node $n \in \mathcal{N}$ if there exists an index i such that $n \in v(\pi_i)$.

A **path-planning problem with visibility constraints and a transit point (PPVT)** is represented by a tuple $\langle \mathcal{D}, s, g, t \rangle$, where $\mathcal{D} = \langle \mathcal{N}, \mathcal{T}, \mathcal{E}, c, v \rangle$ is a domain, $s \in \mathcal{N}$ is the start node, $g \in \mathcal{N}$ is the goal node, $t \in \mathcal{T}$ is the transit point that must be covered. The solution to a PPVT is a path π such that $\pi_1 = s$, $\pi_{|\pi|} = g$, and $\exists i \in \{1, 2, ..., |\pi|\}$, $t \in v(\pi_i)$.

A **path planner** $\mathcal{A}$ takes as input a PPVT and returns a feasible path π for that problem, i.e., $\mathcal{A}(\langle \mathcal{D}, s, g, t \rangle) = \pi$. $\mathcal{A}$ returns Failure when it cannot find a feasible path. For convenience, we define Failure such that it is not equal to itself, i.e., Failure $\neq$ Failure.

We assume that there is an adversary who seeks to deduce the actual transit point $t \in \mathcal{T}$ by observing the trajectory of the agent.

A **Transit Obfuscation Problem** (TOP) is a tuple $\langle \mathcal{D}, s, g, \mathcal{O} \rangle$, where $\mathcal{O}$ describes what the adversary can observe.

We make the following assumptions about the abilities of the adversarial observer (similar to the set of assumptions by [7]):

- **Complete Knowledge about the Domain and Transit Candidates**: The adversary has complete knowledge about the domain $\mathcal{D}$.
- **Full Access to the Planner**: The adversary has full access to and thoroughly understands the agent's planning algorithm.
- **Independence of Inputs**: The adversary can execute the agent's planner with arbitrary input tuples at any time.
- **Observability of Path**: The adversary can immediately observe the path executed by the agent so far.
- **Semi-Honest Adversary**: The adversary is passive, and it does not disturb the action of the agent or gain any additional information beyond what has been specified above.

These are challenging assumptions when trying to conceal the transit point, as the adversary has full information about the mechanism of the path planning algorithm, as well as the ability to rerun/simulate the algorithm many times in order to gain information that might reveal the transit point. When $t = g$, and v is the identity function (the only node visible from a node is itself), this special case of the TOP is similar to the Goal Obfuscation Problem [7, 8]. Unlike the Goal Obfuscation Problem, where the final node of the path always reveals the actual goal, in the TOP, when $t \neq g$, it is possible to have a path where an adversary cannot deduce the actual transit point even after observing the entire trajectory. For example, in Fig. 1, an agent travels from s to g while covering one of $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, where black cells are obstacles. Then, if the agent can see nodes within a radius of one, any feasible path covers all of $\mathcal{T}$ so that an observer cannot infer the true transit.

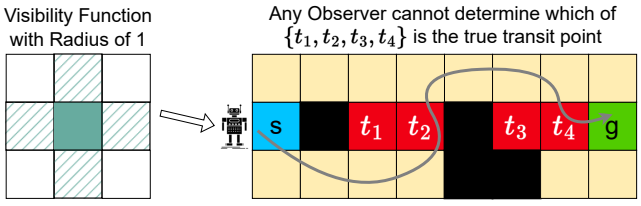

Figure 1: Let nodes within a radius of one be visible. Then, it is not possible for an observer to determine which of t_1, t_2, t_3, t_4 is the true transit point.

3 (k, ℓ, m)-ANONYMITY

Next, we formally define the conditions when a path is anonymized for a transit point, what inputs are anonymizable, and what planners can achieve anonymization.

3.1 Definitions of (k, ℓ, m)-Anonymity

In order for a transit point t to remain private up to the first m steps, even if the adversary has the capabilities enumerated above, it must not be possible to uniquely identify t by executing the planner (possibly many times) and observing the output(s) as well as the internal state of the planner.

This is possible if the route output by a planner is indistinguishable for multiple transit candidate points, including the actual transit point t. For example, if the set of transit candidates $\mathcal{T} = \{t_1, t_2\}$, and $\mathcal{A}(\langle \mathcal{D}, s, g, t_1 \rangle) = \mathcal{A}(\langle \mathcal{D}, s, g, t_2 \rangle) = \pi_{1,2}$, it is indeterminate which of t_1 or t_2 is the true transit point t.

In addition, it is often desirable for the transit candidates to be spread out. For example, if all the transit candidates are close to each other, the adversary may be able to cost-effectively block access to all transit candidates (preventing the agent from covering the transit point). Therefore, it is desirable to be able to disperse the indistinguishable transit candidates in the search space.

Based on this idea, we first define (k, ℓ, m)-Anonymized Paths.

Definition 1 ((k, ℓ, m)-Anonymized Path). *Let $k \in \mathbb{Z}_{>0}$, $\ell \in \mathbb{R}_{\geq 0}$ and $m \in \mathbb{Z}_{>0}$. We say that $\mathcal{A}(\langle \mathcal{D}, s, g, t \rangle)$, a path planned by $\mathcal{A}$ from s to g covering t, where $s \neq t$ and $g \neq t$, is a (k, ℓ, m)-Anonymized Path with respect to t if there exists a set $T \subseteq \{t' | t' \in \mathcal{T}$ and $\mathcal{A}(\langle \mathcal{D}, s, g, t \rangle)|_m = \mathcal{A}(\langle \mathcal{D}, s, g, t' \rangle)|_m\}$ satisfying $|T| \geq k$ and $\min_{(i,j) \in T \times T} d(i, j) \geq \ell$.*

In other words, a path is (k, ℓ, m)-Anonymized when there are at least k transit candidates that result in the same path up to the first m nodes from s to g covering t, and the distance between any pair of nodes within that set is equal to or greater than ℓ. When $m = \infty$, the adversary cannot determine which of those transit candidates is the true t even after observing the entire path.

Second, we define a (k, ℓ, m)-Anonymizable Tuple.

Definition 2 ((k, ℓ, m)-Anonymizable Tuple). *Let $k \in \mathbb{Z}_{>0}$ and $\ell \in \mathbb{R}_{\geq 0}$. Given a domain $\mathcal{D}$, we say that the tuple $\langle \mathcal{D}, s, g, t \rangle$, where $s \in \mathcal{N}$, $g \in \mathcal{N}$, $t \in \mathcal{T}$, $s \neq t$, and $g \neq t$, is a (k, ℓ, m)-Anonymizable Tuple if there exists a path planner $\mathcal{A}$ such that there exists a set $T \subseteq \{t' | t' \in \mathcal{T}$ and $\mathcal{A}(\langle \mathcal{D}, s, g, t \rangle)|_m = \mathcal{A}(\langle \mathcal{D}, s, g, t' \rangle)|_m\}$ satisfying $|T| \geq k$ and $\min_{(i,j) \in T \times T} d(i, j) \geq \ell$.*

The input tuple is (k, ℓ, m)-Anonymizable when at least one planner can output a (k, ℓ, m)-Anonymized Path for this input. Since k is a positive integer, a tuple $\langle \mathcal{D}, s, g, t \rangle$ is not (k, ℓ, m)-Anonymizable Tuple for any k if there exists no path from s to g covering t.

Based on the above definitions, we define (k, ℓ, m)-Anonymity of a path planner $\mathcal{A}$ as follows.

Definition 3 ((k, ℓ, m)-Anonymity). *A path planner $\mathcal{A}$ satisfies (k, ℓ, m)-Anonymity for a domain $\mathcal{D}$ if $\mathcal{A}$ returns a (k, ℓ, m)-Anonymized Path for all (k, ℓ, m)-Anonymizable Tuples in $\mathcal{D}$.*

A path planner with (k, ℓ, m)-Anonymity is guaranteed to return anonymized paths for all anonymizable tuples in $\mathcal{D}$.

We also consider the anonymity of the planner for fixed source and goal locations and define (k, ℓ, m, δ)-Local Anonymity as follows:

Definition 4 ((k, ℓ, m, δ)-Local Anonymity). *Let x be the number of (k, ℓ, m)-Anonymizable Tuples in the domain $\mathcal{D}$ with a fixed source s and goal g. We say that $\mathcal{A}$ satisfies (k, ℓ, m, δ)-Local Anonymity for $\langle \mathcal{D}, s, g \rangle$ if $\mathcal{A}$ returns (k, ℓ, m)-Anonymized Paths for δx or more (k, ℓ, m)-Anonymizable Tuples in $\mathcal{D}$ with s and g.*

A planner $\mathcal{A}$ satisfying (k, ℓ, m)-Anonymity in $\mathcal{D}$ satisfies $(k, \ell, m, 1)$-Local Anonymity for any combination of a start s and a goal g.

3.2 Properties of (k, ℓ, m)-Anonymity

We have identified several important properties about (k, ℓ, m) - Anonymity: equivalence conditions for (k, ℓ, m) - Anonymizable Tuples and Anonymized Paths, path-extensibility, and existence guarantee of a planner achieving (k, ℓ, m)-Anonymity. All omitted proofs, as well as some additional properties can be found in Supp. A [18].

First, the following proposition indicates the necessary and sufficient conditions for a path to be a (k, ℓ, ∞)-Anonymized Path.

Proposition 1 (3C Condition for Output Path). *A path $\pi = \mathcal{A}(\langle \mathcal{D}, s, g, t \rangle)$ is (k, ℓ, ∞)-Anonymized Path iff there exists a set of nodes $T \subseteq \mathcal{T}$ satisfying all of the following:*

1. *Cardinality: $|T| \geq k$*
2. *Cost: $\min_{(i,j) \in T \times T} d(i, j) \geq \ell$*
3. *Coverage: π covers all nodes in T, and $\mathcal{A}$ returns π whenever the transit point belongs to T.*

The similar necessary and sufficient conditions for an input tuple to be (k, ℓ, ∞)-Anonymizable are in Supp. A.2 [18].

Next, the coverage condition of Prop. 1 leads to the computational complexity of planning a (k, ℓ, ∞)-Anonymized Path.

Theorem 1 (Complexity). *Finding a (k, ℓ, m)-Anonymized Path for the given tuple is NP-Hard.*

Proof of Theorem 1. Finding a path that covers all nodes in the given set of nodes is a generalization of WRP, which is NP-Hard [14] (see Sec. 4.3) □

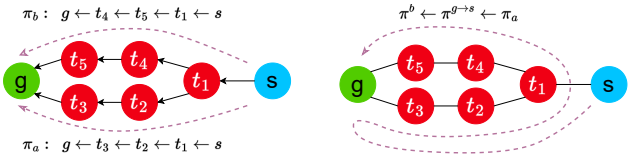

Figure 2: Directed vs. Undirected: While all tuples are $(3, 1, \infty)$-Anonymizable Tuples, planning $(3, 1, \infty)$-Anonymized Path for every transit point is impossible in the directed (left) case.

If the domain consists of an undirected graph, we can ensure the existence of a path planner that satisfies (k, ℓ, m)-Anonymity.

Theorem 2 (Existence of a Satisfying Path Planner). *If every edge in the domain $\mathcal{D}$ is undirected, meaning that $\forall (i, j) \in \mathcal{N} \times \mathcal{N}$, $(i, j) \in \mathcal{E} \Rightarrow (j, i) \in \mathcal{E}$, there exists a path planner $\mathcal{A}$ that satisfies (k, ℓ, m)-Anonymity for any given k, ℓ and m.*

To prove this, we use below which states that extending a (k, ℓ, m)-Anonymized Path preserves the same level of anonymity.

Lemma 1 (Path-Extension). *Let $\pi = A(\langle \mathcal{D}, s, g, t \rangle)$ be a (k, ℓ, m)-Anonymized Path with respect to t, $\pi_{s' \to s}$ be a path, constructed independently of t, from s' to s, and $\pi_{g \to g'}$ be an arbitrary path from g to g'. Then, $\pi_{s' \to s} \circ \pi \circ \pi_{g \to g'}$ is also a (k, ℓ, m)-Anonymized Path with respect to t.*

Proof of Theorem 2. Let $\hat{T}_{s,g} = \{t \mid t \in \mathcal{T}, \langle D, s, g, t \rangle$ is (k, ℓ, m) -Anonymizable Tuple$\} = \{t_1, t_2, ..., t_x\}$, and $\hat{\pi}^{t_i}_{s \to g}$ be the (k, ℓ, m)-Anonymized Path for $\langle D, s, g, t_i \rangle$. If $|\hat{T}_{s,g}| \geq 1$ and all edges in $\mathcal{D}$ are undirected, there exists $\pi_{g \to s}$, a path from g to s. By Lemma 1, we have that $\hat{\pi}_{s \to g} = (\|_{i=1}^{x-1} \hat{\pi}^{t_i}_{s \to g} \circ \pi_{g \to s}) \circ \hat{\pi}^{t_x}_{s \to g}$ is a (k, ℓ, m)-Anonymized Path with respect to all transit nodes in $\hat{T}_{s,g}$.

Now, consider a path planner $\mathcal{A}$ such that $\mathcal{A}(\langle D, s, g, t \rangle) = \hat{\pi}_{s \to g}$ if $|\hat{T}_{s,g}| \geq 1$ and $\mathcal{A}(\langle D, s, g, t \rangle) =$ Failure otherwise. It is evident that $\mathcal{A}$ satisfies (k, ℓ, m)-Anonymity: □

Although Theorem. 2 guarantees that there exists a path planner with (k, ℓ, m)-Anonymity for any undirected graph, such a guarantee is not possible when the edges are directed. A counterexample is shown in the left-side of Fig. 2, where there are two possible paths from s to g; $\pi_a = (s, t_1, t_2, t_3, g)$ or $\pi_b = (s, t_1, t_4, t_5, g)$. Thus, if we assume that the costs of edges are all one, all input tuples are $(3, 1, \infty)$-Anonymizable Tuples. Let $\pi_i = \mathcal{A}(\langle \mathcal{D}, s, g, t_i \rangle)$. Clearly, any path planner $\mathcal{A}$ should satisfy $\pi_a = \pi_2 = \pi_3$ and $\pi_b = \pi_4 = \pi_5$. Then, if $\pi_a = \pi_1$, π_a is a $(3, 1, \infty)$-Anonymized Path but π_b is not a $(3, 1, \infty)$-Anonymized Path. Likewise, if $\pi_b = \pi_1$, π_b is a $(3, 1, \infty)$-Anonymized Path but π_a is not a $(3, 1, \infty)$-Anonymized Path. On the other hand, if all edges are undirected, we can construct a $(3, 1, \infty)$-Anonymized Path from π_a and π_b by concatenating them.

4 PBP: PARTITIONING-BASED PLANNER FOR (k, ℓ, ∞)-ANONYMITY

We now propose an algorithm for (k, ℓ, ∞)-Anonymity. First, note that *if we disregard path cost*, then in principle, a relatively straightforward approach to achieve (k, ℓ, ∞)-Anonymity would be to return some path which covers all transit candidates. However, a practical algorithm for Transit Obfuscation needs to effectively trade

off the privacy objective vs. path costs, on average, overall (s, g, t) tuples of interest. We first propose objectives that express this trade-off and then propose a partitioning-based algorithm which seeks a solution which optimizes this objective. Proofs are in Supp. B [18].

4.1 Objectives

We define two metrics, Anonymized Path Ratio (APR) and Mean Anonymization Cost (MAC), to evaluate the performance of planners that satisfy (k, ℓ, m)-Anonymity. Let $\mathscr{T}_+$ be the set $\{t | t \in \mathscr{T}, \mathcal{A}(\langle \mathcal{D}, s, g, t\rangle)$ is a (k, ℓ, m)-Anonymized Path$\}$ for the fixed $\mathcal{D}$, s, and g. APR is defined as follows:

Definition 5 (Anonymized Path Ratio (APR)).

$$APR(\mathcal{A}, D, s, g) = \frac{|\mathscr{T}_+|}{\#Coverable\ Transit\ Nodes}$$

where *#Coverable Transit Nodes* denotes the number of transit nodes within $\mathscr{T}$ such that there exists a path from s to g covering $t \in \mathscr{T}$. APR is equivalent to the lower bound of δ, and a larger APR is desirable.

Next, inspired by deception cost [12], a metric for goal obfuscation, we define the MAC metric:

Definition 6 (Mean Anonymization Cost (MAC)).

$$MAC(\mathcal{A}, D, s, g) = \sum_{t \in \mathscr{T}_+} \frac{cost(\mathcal{A}(\langle \mathcal{D}, s, g, t\rangle)) - cost(\pi^{t*})}{|\mathscr{T}_+|\ \ cost(\pi^{t*}))}$$

where π^{t*} is the shortest path from s to g covering t. MAC shows the cost of anonymizing the transit points.

4.2 Pbp: Partitioning-based Planner

Algorithm 1 Partitioning-based Planner (Pbp)

Input: Tuple $\langle \mathcal{D}, s, g, t\rangle$ and privacy parameters (k, ℓ)
Output: a path π from s to g covering t

1: /* Pre-Processing **independent of** t*/
2: Generate a partition of $\mathscr{T}, T_1, T_2, ..., T_\Phi$ such that $\bigcup_{\phi=1}^{\Phi} T_\phi = \mathscr{T}$, any T_ϕ except T_Φ meets all conditions of Prop. 1, and any pair are disjoint.
3: /* End of Pre-Processing */
4: **if** t belongs to T_Φ **then return** Failure
5: $T \leftarrow$ the partition that contains t
6: $\pi^*_T \leftarrow$ shortest path from s to g while covering T
7: **return** π^*_T

We now propose the Partitioning-based Planner (Pbp), a practical algorithm that seeks to achieve (k, ℓ, ∞)-Anonymity while optimizing the above objectives.

Pbp is composed of two phases: (1) the Partitioning/Pre-Processing phase and (2) the path query phase. In the Partitioning/Pre-Processing phase, the algorithm searches for a partition of all transit candidates, denoted as $T_1, T_2, ..., T_\Phi$, such that (a) each subset except the final T_Φ satisfies the 3C Conditions described in Prop. 1, ensuring that each subset covers the required nodes to achieve the desired level of anonymity, and (b) the objectives are optimized. The set T_Φ consists of transit candidates that the planner cannot anonymize. This phase only needs to be executed once for each domain.

In the path query phase, given a specific s, g, and t, the algorithm finds the shortest path π from the source node s to the goal node g while covering all the nodes assigned to the subset (computed above in the Partitioning phase) that includes the target node t.

By utilizing this approach, Pbp can effectively plan a (k, ℓ, ∞)-Anonymized Path, ensuring the privacy requirements are met while efficiently navigating from the source to the destination. If the obtained partition is perfect, Pbp satisfies (k, ℓ, ∞)-Anonymity.

Theorem 3 (Completeness). *Let all edges in $\mathcal{D}$ be undirected. Then, if $\sum_{\phi=1}^{\Phi-1} |T_\phi|$ is maximized for any pair of s and g, Alg. 1 satisfies (k, ℓ, ∞)-Anonymity.*

4.3 WRP With Targets (WRPT)

A key building block for the Pbp algorithm is a search algorithm for finding the minimum cost path π from s to g, which covers all of the nodes in a set of nodes. We call this the Watchman Route Problem with Targets (WRPT). The WRPT corresponds to the subproblem solved by the path query phase of Pbp in Alg. 1, line 6. The WRPT is also used in the Partitioning phase when evaluating candidate partitionings (Alg. 2, line 18).

The WRPT is a variant of the Watchman Route Problem (WRP) [15], The differences between the WRP and WRPT are: (1) WRP does not have a specified goal node, while the WRPT has a specific goal g, and (2) the objective of the WRP is to cover all nodes in the graph, while the WRPT seeks to cover some subset ψ of nodes in the graph. Since WRP was shown to be NP-Hard [15], the WRPT is clearly NP-Hard. Recent work has studied heuristic search-based algorithms to solve WRP [15].

Following Skyler et al. [15], we use an A* search for the WRPT. We define a state for the search as a tuple $\langle n, \mathcal{U}\rangle$, where $n \in \mathcal{N}$ represents the current location, and $\mathcal{U} \subseteq \mathcal{N}$ represents the set of uncovered nodes. The initial state is $\langle s, \psi \setminus v(s)\rangle$, and the final is $\langle g, \emptyset\rangle$. Expanding a state $\langle n, \mathcal{U}\rangle$ involves moving from n to one of its neighboring nodes n' and updating the set of uncovered nodes $\mathcal{U}$ to $\mathcal{U} \setminus v(n')$. The cost of this expansion equals $c(n, n')$.

To make the search more efficient, we propose Tunnel Heuristic, which is based on the Singleton Heuristic for the WRP [15]. The Tunnel Heuristic value h_{tunnel} is computed as follows:

$$h_{tunnel}(\langle n, \mathcal{U}\rangle) = \begin{cases} (\max\limits_{u \in \mathcal{U}} \min\limits_{q \in v^{-1}(u)} d(n, q)) & \\ +(\min\limits_{u \in \mathcal{U}} \min\limits_{q \in v^{-1}(u)} d(q, g)) & \text{if } \mathcal{U} \neq \emptyset \\ d(n, g) & \text{otherwise} \end{cases}$$

where the function $v^{-1}(n) : \mathcal{N} \rightarrow 2^{\mathcal{N}}$ takes a node and returns the set of nodes from which n is observable. The heuristic h_{tunnel} is admissible since the agent must travel to one of the nodes in $v^{-1}(u)$ to observe an uncovered node u and then proceed to the goal g after covering all nodes in $\mathcal{U}$.

4.4 Searching for a Partitioning

Alg. 1 requires an algorithm that generates a partition of the set of nodes. Let $\Psi_+ \subseteq \Psi$ be the largest subset of Ψ whose elements all satisfy the conditions of Prop. 1. We denote the sum of the

cardinalities of the subsets in Ψ_+ as $|ap|$, and the *MAC* corresponding to Ψ_+ as *mac*. Specifically, *mac* is $\sum_{\psi\in\Psi_+} ac(\psi)/|ap|$, where $ac(\psi) = \sum_{k\in\psi}(\pi^*_\psi - \pi^{k*})/(\pi^{k*})$, where π^{k*} denotes the minimal cost path from s to g covering k, and π^*_ψ denotes the minimal cost path from s to g while covering all nodes within ψ, i.e., the solution to a WRPT which covers ψ. We seek a partitioning which first prioritizes maximizing $|ap|$, then minimizes *mac*, i.e., a partitioning which anonymizes as many transit nodes as possible while minimizing the average cost of the anonymized paths.

Algorithm 2 Merge-BB Partitioning

Input: Tuple $\langle \mathcal{D}, s, g, t\rangle$ and privacy parameters (k, ℓ)
Output: The best partition Ψ^* of $\mathscr{T}$

1: $|ap|^* \leftarrow 0$, $mac^* \leftarrow \infty$, $\Psi^* = \emptyset$
2: **function** Merge_BB_Search(Ψ)
3: $\Psi_+ \leftarrow \{\psi | \psi \in \Psi$ s.t. ψ satisfies all conditions of Prop. 1 $\}$
4: $|ap| \leftarrow \sum_{\psi\in\Psi_+} |\psi|$, $mac \leftarrow \sum_{\psi\in\Psi_+} ac(\psi)/|ap|$
5: **if** $(|ap| > |ap|^*)$ or $(|ap| = |ap|^*$ and $mac < mac^*)$ **then**
6: $|ap|^* \leftarrow |ap|$, $mac^* \leftarrow mac$, $\Psi^* \leftarrow \Psi$
7: **if** $(|\Psi| = 1)$ or $(|ap| = \sum_{\psi\in\Psi} |\psi|)$ or $(|ap|^* = \sum_{\psi\in\Psi} |\psi|$ and $mac \geq mac^*)$ **then return** True
8: **for** $(i, j) \in MergeOrder(\Psi)$ **do**
9: **if** Prunable(ψ_i, ψ_j) **then** Continue
10: $\pi^*_{\psi_i\cup\psi_j} \leftarrow$ shortest path from s to g, covering $\psi_i \cup \psi_j$
11: **if** $\pi^*_{\psi_i}$ is not found **then** Continue
12: $\Psi' \leftarrow \Psi \setminus \{\psi_i, \psi_j\} \cup \{\psi_i \cup \psi_j\}$
13: Merge_BB_Search(Ψ')
14:
15: **for** $i \in \mathscr{T}$ **do**
16: **if** $\exists u \in v(i)$ u is reachable from s and $\exists u \in v(i)$ g is reachable from u **then**
17: $\psi_i \leftarrow \{i\}$
18: $\pi^*_{\psi_i} \leftarrow$ shortest path from s to g, covering t
19: Merge_BB_Search($\{\psi_1, \psi_2, ...\}$)
20: **return** $\Psi^*_+ \cup \{\bigcup \Psi^* \setminus \Psi^*_+\}$

4.4.1 Merge-based Branch-and-Bound. One practical partitioning algorithm is Merge-based Branch-and-Bound Partitioning (Merge-BB). It initially assigns each node to its own separate partition. It removes any node without a valid path from s to g while covering that node. This pruning step involves calculating all pair-wise shortest paths on the node set N, which can be done efficiently within a reasonable amount of time. The algorithm then performs a recursive branch-and-bound search which considers all possible combinations of merges of these partitions.

The termination condition for this recursive search is implemented in Line 7: Return True when (1) the partition Ψ contains only one subset, or (2) $|ap|$ reaches the upper bound, or (3) the current best partition Ψ^* has an optimal $|ap|$, and the *mac* of Ψ is not better than mac^*. The third termination condition is based on the observation that the cost of the optimal path covering all nodes in the union of ψ_i and ψ_j is always equal to or greater than both of the costs of the optimal paths covering all nodes in ψ_i and ψ_j.

Alg. 2 explores all potential merges and returns a partition that maximizes the number of anonymized transit points while minimizing the average cost of anonymized paths.

Proposition 2 (Optimality). *Alg. 1 using Alg 2 achieves the largest APR and also has the minimum MAC among planners with the largest APR for any combination of $\mathcal{D}$, s, and g.*

Since Theorem 2 tells that there exists a planner satisfying (k, ℓ, ∞)-Anonymity for an undirected graph, which means that it can anonymize all (k, ℓ, ∞) - Transit Anonymizable Tuples, combining Theorem 3 and Prop. 2 immediately yields the guarantee that Alg. 1 with Alg. 2 satisfies (k, ℓ, ∞)-Anonymity.

We also implemented the following enhancements.

Merge Ordering Strategies. The order in which partitions are merged in Alg. 2 by the recursive enumeration is determined by a call to the MergeOrder function in line 8, which returns the list of all pairs of candidate subsets to merge, sorted according to some merge ordering criterion. One simple strategy is Random, which simply returns a randomly shuffled list of the pairs of partitions. Another ordering strategy, CostAsc, sorts the pairs to be merged in ascending order of a heuristic cost function. We use $max(cost(\pi_{\psi_i}), cost(\pi_{\psi_j}))\times(|\psi_i|+|\psi_j|)$ as the cost of a pair (ψ_i, ψ_j), where the first term is the lower bound of the covering path of the merged partition, and the second term is the number of transit candidates assigned to the merged partition. CostAsc helps the planner find a better solution earlier, leading to more upper/lower bound-based pruning.

Pruning Criteria. To determine whether we need to try merging (ψ_i, ψ_j), Alg. 2, line 9 calls Prunable (Alg.3). If both ψ_i and ψ_j already satisfy all the conditions stated in Thm.1, the merge is pruned because it would increase the cost of a path covering all the nodes within the set (Alg. 3, Line 2). Furthermore, suppose we have already found a satisfactory solution for anonymizing all possible tuples. In that case, we can establish an upper bound for the path cost covering the union of ψ_i and ψ_j to surpass the current best satisfying solution and prune based on this bound (Alg. 3, Line 3). Specifically, we denote $\hat{ac}$ as the upper bound for the cost that $\pi_{\psi_i\cup\psi_j}$ must meet to improve upon the best ac^*, and it is calculated as $|ap|^*mac^* - |ap|mac + ac(\pi_{\psi_i}) + ac(\pi_{\psi_j})$. To estimate the cost of a path that covers all the nodes within the union of ψ_i and ψ_j, we use $\max(cost(\pi^*_{\psi_i}), cost(\pi^*_{\psi_j}))$, which is the lower-bound of $cost(\pi^*_{\pi_i\cup\pi_j})$. Finally, we check the minimum distance between a node in ψ_i and ψ_j. If that distance is less than ℓ, we prune this merge, as any set containing the union of ψ_i and ψ_j would also violate this condition (Alg. 3, Line 7).

5 PLANNERS FOR m-BOUNDED (k, l, m)-ANONYMITY

We now propose three planners that output (k, ℓ, m)-Anonymized Paths with $m < \infty$, which means that the adversary cannot identify which node is the transit point until observing more than m nodes. If m is less than the length of the output path, the adversary might be able to identify the true transit point after obtaining the $m+1$st and later nodes. Proofs for this section can be found in Supp. C [18].

Algorithm 3 Prunable

if Both ψ_i and ψ_j meets all conditions in Prop. 1 **then**
  return True
if $\min_{(x,y)\in\psi_i\times\psi_j} d(x,y) < \ell$ **then return** True
if $|ap|^* = \sum_{\psi\in\Psi} |\psi|$ **then**
  $\hat{ac} = |ap|^* mac^* - |ap| mac + ac(\pi_{\psi_i}) + ac(\pi_{\psi_j})$
  $\bar{ac} = \sum_{k\in\psi_i\cup\psi_j} \frac{\max(cost(\pi^*_{\psi_i}), cost(\pi^*_{\psi_j})) - cost(\pi^{k*})}{cost(\pi^{k*})}$
  if $\bar{ac} \geq \hat{ac}$ **then return** True
return False

Random-Walk-based Planner (Rbp). The first algorithm is a Random-Walk-based Planner (Rbp), which randomly selects the path's first m nodes. Specifically, the planner selects the i-th node ($2 \leq i \leq m$) of the path randomly from the neighbors of the $(i-1)$-th node, where $\pi_1 = s$. Subsequently, the planner guides the agent's movement from π_m to t and finally to g utilizing the shortest paths available. $\pi|_m$ is the same for all transit candidates, and the planner outputs Failure if there is no feasible path. This planner archives (k, ℓ, m)-Anonymity with finite m for undirected graphs.

Proposition 3. *If all edges in $\mathcal{E}$ are undirected, Random-Walk-based Planner satisfies (k, ℓ, m)-Anonymity with $m < \infty$.*

m-Pbp. The second planner is m-Pbp, which is an extension of Pbp. m-Pbp returns $\pi_{Pbp}|_m \circ \pi^*_{ua}$, where $\pi_{Pbp}|_m$ is the output path up to the m-th node planned by Pbp, and π^*_{ua} is the unanonymized shortest path from the last node of $\pi_{Pbp}|_m$ to the goal while covering t if $\pi_{Pbp}|_m$ does not cover t. The anonymity of m-Pbp relies on the anonymity of Pbp.

Proposition 4. *If Pbp satisfies (k, ℓ, ∞)-Anonymity for the given domain $\mathcal{D}$, m-Pbp satisfies (k, ℓ, m)-Anonymity for that domain $\mathcal{D}$.*

Clustering-based Planner (Cbp). The third m-bounded planner is a Clustering-based Planner (Cbp), which first applies k-means-like clustering to the transit candidates and then returns the concatenation of paths from s to the centroid of the cluster corresponding to t and from the centroid to g. Following Wasserman and Faust [20], we call a node $\sigma \in \mathcal{N}$ that minimizes the maximum distance to cover a node within a set of nodes $\mathcal{U} \subseteq \mathcal{N}$ the *centroid* of $\mathcal{U}$, i.e., $\sigma = \arg\min_{n\in\mathcal{N}} \max_{u\in\mathcal{U}} \min_{w\in v^{-1}(u)} d(n, w)$. Like k-means clustering [1], Cbp iteratively updates the assignment of each transit candidate to minimize the distance between each candidate and the centroid of their respective cluster. After the assignments stabilize, Cbp repeatedly merges a cluster with cardinality less than k with the nearest cluster until each cluster has k or more nodes. Here, we use the shortest distance from the centroid of the i-th cluster to the centroid of the j-th cluster as the distance from the i-th cluster to the j-th cluster. Then, Cbp checks $|\pi_{s\to\sigma_t}|$, the length of the shortest path from s to the centroid of the cluster containing the true transit node t. If it exceeds m, Cbp assigns the first m nodes from $\pi_{s\to\sigma_t}$, as π^1. Otherwise, Cbp appends a randomly generated path as padding (Line 12-15 in Algorithm 4) and assigns the extended path to π^1. Finally, Cbp computes π^2, the shortest path from the last node of π^1 to g, covering t, and returns $\pi^1 \circ \pi^2$. The sequence of the first m nodes planned by Cbp is the same for all transit points belonging to the same cluster. Cbp satisfies the following.

Proposition 5. *Let all edges in $\mathcal{D}$ be undirected. If for any $t \in \mathcal{T}$, there exists a path from s to g while covering t, Cbp satisfies $(k, 0, m)$-Anonymity.*

Algorithm 4 Clustering-based Planner (Cbp)

1: Randomly assign the transit candidates to $[|\mathcal{T}|/k]$ clusters s.t. the number of nodes in each cluster is equal to or more than k
2: **while** True **do**
3: **for** each transit candidate $t_i \in \mathcal{T}$ **do**
4: Compute the distance to each centroid
5: Assign t_i to the cluster with the nearest centroid.
6: **for** each cluster **do**
7: Recompute the centroid for each cluster;
8: **if** the assignment does not change **then** Break
9: **while** There exists a cluster consisting of less than k nodes **do**
10: **for** each cluster consisting of less than k nodes **do**
11: Assign all elements of this cluster to the nearest cluster
12: $\sigma_t \leftarrow$ the centroid of the cluster containing t
13: $\pi^*_{s\to\sigma_t} \leftarrow$ the shortest path from s to σ_t
14: **if** $|\pi^*_{s\to\sigma_t}| \geq m$ **then** $\pi^1 \leftarrow \pi^*_{s\to\sigma_t}|_m$
15: **else**
16: $r = s$, $\pi_{s\to r} = s$
17: **while** $|\pi_{s\to r}| + |\pi^*_{r\to\sigma_t}| < m$ **do**
18: $r \leftarrow$ Randomly pick the neighbour of r
19: Append r to the tail of $\pi_{s\to r}$
20: $\pi^1 \leftarrow \pi_{s\to r} \circ \pi^*_{r\to\sigma_t}$
21: $\pi^2 \leftarrow$ shortest path from $\pi^1_{|\pi^1|}$ to g while covering t
22: **return** $\pi^1 \circ \pi^2$

6 EXPERIMENTS

Benchmarks and Settings. We evaluate the performance of Pbp on six 2D grid world benchmark instances from the Moving AI pathfinding benchmark set [16]: den101d, den201d, lak102d, lak510d, orz000d, and orz201d. We randomly select 5 pairs of start and goal points for each benchmark. The number of transit candidates, $|\mathcal{T}|$, is 8, 12, and 16, and we randomly select $\mathcal{T}$ from $\mathcal{N}$ for each problem. We use a visibility function where nodes within a range less than or equal to distance r are covered (for r =0,2,10). All experiments used 4-way unit cost movement.

All algorithms were implemented in C++, and experiments were run on an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz and 125GB of RAM, running Ubuntu 22.04.2 LTS. A time limit of 300 seconds/instance was used.

6.1 Evaluation of Pbp with $m = \infty$

We evaluate Pbp using Merge-BB and DF-BB partitioning strategies, two merge orders (Random and CostAsc), and two WRTP heuristics: the blind heuristic (equivalent to breadth-first search) and the Tunnel heuristic. We also evaluate two baseline partitionings:

Baseline #1: Naive. This generates a partitioning by randomly splitting the transit candidates into pairs of 2 nodes. This approach satisfies $(2, 1, \infty)$-Anonymity for this class of undirected grid maps.

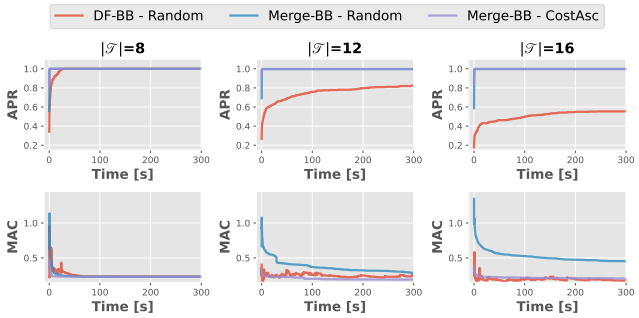

Figure 3: Convergence of ARP and MAC. Mege-BB with CostAsc shows the best performance.

Baseline #2: Depth-First Branch-and-Bound (DF-BB) partitioning. DF-BB starts with all nodes unassigned and then constructs a partitioning by assigning one unassigned node to an existing subset of Φ or a new subset (details in Supp. D [18]).

For the privacy parameters, we set k to 2 and 3 and ℓ to 1 and 10.

Results. Tab. 1 shows the performance of the Partitioning-based Planner when $k = 2, \ell = 1$. We report coverage, APR, MAC, and execution time. Coverage is the percentage of problems on which Pbp completed the search (found the optimal solution and proved its optimality) within the time limit. We report mean MAC for the configurations which found satisfying solutions (ARP=1) for all instances within the time limit. Total Time denotes the average execution time for the instances where Merge-BB with the blind heuristic completed the search for $|\mathscr{T}|$ of 8 and 12.

Tab. 2 shows the mean and standard deviation of the MAC of Merge-BB divided by the MAC of Naive, showing over 50% improvement of Merge-BB with CostAsc.

From Tab. 1-2, we observe that: (1) Pbp (Merge-BB) consistently results in better MAC than the Naive baseline, showing that searching for an optimal partition achieves significantly better path costs than a naive partitioning. (2) Merge-BB has significantly higher coverage than DF-BB, showing that the merge-based approach is a more efficient strategy. (3) Overall, combining Merge-BB, CostAsc, and Tunnel gives the best performance.

Fig. 3 depicts the changes over time in ARP and MAC when $k = 2$, $\ell = 1$. All combinations use h_{tunnel}. Mege-BB archives higher APR faster, and CostAsc can find solutions with lower MAC earlier.

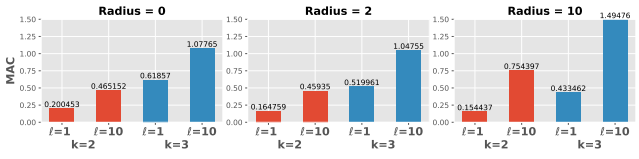

Figure 4: Impact of k, ℓ, and radius r. Larger k and ℓ increase MAC. The impact of r is not monotonic.

Fig. 4 shows MAC for each combination of (k, ℓ, m) and r when using Merge-BB with CostAsc and h_{tunnel}. Larger k and ℓ result in worse MAC. The correlation between MAC and r is not monotonic since larger r allows the agent to cover nodes with less movement, decreasing both the numerator and denominator of MAC.

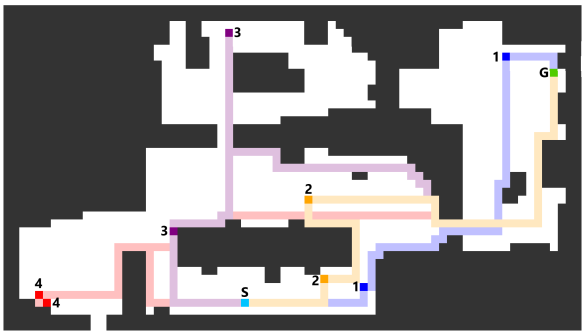

Figure 5: Example of $(2, 1, \infty)$-Anonnymized Paths.

FIg. 5 shows an example of optimal $(2, 1, \infty)$-Anonnymized Paths in *den101* obtained by Pbp (Merge-BB). The cyan (S) and green (G) cells are the source and goal, respectively. The transit candidate belonging to the same subset ($1 \sim 4$) has the same color, while its corresponding path is colored in a lighter color. More qualitative examples can be found in Supp. E.

6.2 Evaluation with bounded m

We compared the three m-bounded anonymity planners, m-Pbp, Rbp, and Cbp, for various values of m and k. For each problem, we set m such that $m/|\pi^*|$ is [0.1, 0.3, 0.5, 1.0, 5.0, 10.0], where π^* is the length of the shortest path for that problem. k is [2, 3, 5]. The remaining parameters are kept constant: $\ell = 1$, $|\mathscr{T}| = 8$, and the heuristic function is h_{tunnel}.

Fig. 6 shows that as m increases, the MAC of Rbp and Cbp exhibits exponentia growth (y-axis is log scale), while m-Pbp's performance converges towards that of Pbp. m-Pbp aims to cover all nodes within the same partition, while Cbp strives to move towards the centroid of the cluster. Thus, when m is small enough that Cbp doesn't require appending a random-walking path, the MAC of Cbp surpasses that of m-Pbp. However, m-Pbp exhibits superior performance compared to the other methods for larger values of m.

Tab. 3 shows the execution time of each planner. Cbp is faster than m-Pbp. Larger m makes Cbp faster because the path after the m-th node tends to be shorter, reducing the runtime of the search in Line 21 in Alg. 4. However, this effect decreases if m is too large for Cbp to need additional time to append random nodes.

7 RELATED WORK

k-Anonymity. k-anonymity is a fundamental concept in data privacy and anonymization that aims to safeguard individual identities in a dataset while preserving its utility [17, 19]. It seeks to render each record in a dataset indistinguishable from at least k-1 other records, i.e., each individual's data is grouped with a minimum of k-1 other individuals with similar attributes. This grouping makes it difficult to identify a specific individual within the group.

Obfuscation. Some existing methods for goal obfuscation leverage concepts similar to k-anonymity. For instance, Kulkarni et al.

Table 1: Comparison of each combination of partitioning, MergeOrder, and heuristic functions on coverage, the number of evaluated partitions, and the total execution time. Merge-BB with CostAsc using Tunnel achieved the best performance.

		$\lvert\mathscr{T}\rvert$	8	12	16	8	12	16	8	12	16	8	12	16
Planner	MergeOrder	Heuristic	Coverage [%]			Total Time [s]			APR (higher=better)			MAC (lower=better)		
Naive		Blind	n/a	n/a	n/a	<1	<1	-	1.00	1.00	1.00	0.714	0.434	0.582
		Tunnel	n/a	n/a	n/a	<1	<1	-	1.00	1.00	1.00	0.714	0.434	0.582
DF-BB		Blind	77	0	0	19	-	-	1.00	0.672	0.439	0.232	-	-
		Tunnel	100	20	0	2	92	-	1.00	0.828	0.558	0.229	-	-
Merge-BB	Random	Blind	74	7	0	35	185	-	1.00	1.00	1.00	0.232	0.418	0.561
		Tunnel	100	27	0	3	13	-	1.00	1.00	1.00	0.229	0.287	0.452
	CostAsc	Blind	77	7	0	10	80	-	1.00	1.00	1.00	0.231	0.222	0.223
		Tunnel	100	47	0	1	5	-	1.00	1.00	1.00	0.229	0.190	0.205

Table 2: Ratio of Merge-BB's MAC to Naive's MAC. Merge-BB reduces the MAC of a satisfying solution by more than 50%.

MergeOrder	Heuristic	$\lvert\mathscr{T}\rvert = 8$	$\lvert\mathscr{T}\rvert = 12$	$\lvert\mathscr{T}\rvert = 16$
Random	Blind	0.425 (±0.227)	0.679 (±0.380)	0.937 (±0.716)
	Tunnel	0.423 (±0.226)	0.462 (±0.347)	0.706 (±0.723)
CostAsc	Blind	0.435 (±0.237)	0.350 (±0.160)	0.378 (±0.189)
	Tunnel	0.423 (±0.226)	0.313 (±0.140)	0.349 (±0.188)

Table 3: Comparison of the runtime ([s]) for each planner. Rbp and Cbp show better scalability compared to m-Pbp.

$m/\lvert\pi^*\rvert$	0.1	0.3	0.5	1.0	5.0	10.0
Rbp	0.006	0.006	0.006	0.006	0.006	0.006
Cbp	0.098	0.085	0.046	0.009	0.006	0.006
m-Pbp	63.985	63.985	63.985	63.985	63.985	63.985

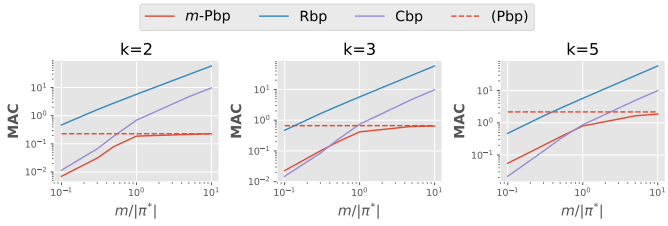

Figure 6: Impact of k and m on MAC. While MAC of Rbp increases linearly with respect to m, MAC of m-Pbp converges to MAC of Pbp.

[7, 8] define a secure path as one where k different goals result in the same path, thereby making it difficult for an observer to discern the true purpose among these k nodes. Another example is *Dissimulation* proposed in Masters and Sardina [11], where the true goal is considered to be obfuscated if there exist other nodes that look like the goal equally or more than the real goal. While our work refrains from making any assumptions regarding how the observer deduces the agent's intention, some studies [9–11, 13] model the inference process of the observer and devise obfuscation techniques tailored to these models. However, approaches based on such models do not provide a guarantee of security against adversaries who do not adhere to the model assumptions.

8 CONCLUSION

This paper introduced the Transit Obfuscation Problem and proposed novel techniques to address this challenge. We introduced (k, ℓ, m)-Anonymity as a measure of concealment achieved by a path planner. We proposed a Pbp, a partitioning based algorithm to achieve (k, ℓ, ∞)-Anonymity, and evaluated its performance on 2D grid maps with obstacles. We showed that Pbp with a merge-based branch-and-bound strategy significantly outperforms baseline partitioning approaches.

Although we showed that Merge-BB with the APR and MAC objectives is a viable approach to partitioning, a complete branch-and-bound search to find and prove the optimality of a solution poses a scalability challenge. For example, although our current implementation of Pbp can find solutions for $\lvert\mathscr{T}\rvert = 16$ (Tab. 1), it can not complete the search and prove optimality within the 300 sec. limit. Search algorithms finding good partitionings quickly without an optimality guarantee (e.g., local search/metaheuristics) are a direction for future work.

We also investigated algorithms for m-bounded anonymity ($m < \infty$). We showed that while m-Pbp yielded the best MAC scores, Cbp offers fairly good MAC scores but runs much faster. Future work will investigate additional approaches to trading off scalability vs. solution quality.

Finally, while this work focused on a single agent and single adversary in a static environment, an extension of our proposed techniques to more complex scenarios, such as multi-agent systems and dynamic environments, is another direction for future work.

REFERENCES

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* 9, 8 (2020), 1295.

[2] Tathagata Chakraborti, Anagha Kulkarni, Sarath Sreedharan, David E Smith, and Subbarao Kambhampati. 2019. Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior. In *Proceedings of the international conference on automated planning and scheduling*, Vol. 29. 86–96.

[3] Rui Chen, Benjamin CM Fung, Bipin C Desai, and Nériah M Sossou. 2012. Differentially private transit data publication: a case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 213–221.

[4] Joao Dias, Ruth Aylett, Ana Paiva, and Henrique Reis. 2013. The great deceivers: Virtual agents and believable lies. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, Vol. 35.

[5] Saeede Enayati, Dennis L. Goeckel, Amir Houmansadr, and Hossein Pishro-Nik. 2022. Privacy-Preserving Path-Planning for UAVs. In *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. 1–6. https://doi.org/10.1109/ISNCC55209.2022.9851770

[6] Sarah Keren, Avigdor Gal, and Erez Karpas. 2016. Privacy Preserving Plans in Partially Observable Environments.. In *IJCAI*. 3170–3176.

[7] Anagha Kulkarni, Matthew Klenk, Shantanu Rane, and Hamed Soroush. 2018. Resource bounded secure goal obfuscation. In *AAAI Fall Symposium on Integrating Planning, Diagnosis and Causal Reasoning*.

[8] Anagha Kulkarni, Siddharth Srivastava, and Subbarao Kambhampati. 2019. A unified framework for planning in adversarial and cooperative environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2479–2487.

[9] Alan Lewis and Tim Miller. 2023. Deceptive Reinforcement Learning in Model-Free Domains. *arXiv preprint arXiv:2303.10838* (2023).

[10] Junren Luo, Wanpeng Zhang, Fengtao Xiang, and Su Jiongming. 2019. Intention Obfuscated Adversarial Deceptive Path Recommendation for UGV Patrol Maneuver. 206–211. https://doi.org/10.1109/IHMSC.2019.00055

[11] Peta Masters and Sebastian Sardina. 2017. Deceptive Path-Planning.. In *IJCAI*. 4368–4375.

[12] Adrian Price, Ramon Fraga Pereira, Peta Masters, and Mor Vered. 2023. Domain-Independent Deceptive Planning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 95–103.

[13] Yagiz Savas, Christos K Verginis, and Ufuk Topcu. 2022. Deceptive decision-making under uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 5332–5340.

[14] Shawn Seiref, Tamir Jaffey, Margarita Lopatin, and Ariel Felner. 2020. Solving the watchman route problem on a grid with heuristic search. In *Proceedings of the international conference on automated planning and scheduling*, Vol. 30. 249–257.

[15] Shawn Skyler, Dor Atzmon, Tamir Yaffe, and Ariel Felner. 2022. Solving the Watchman Route Problem with Heuristic Search. *J. Artif. Intell. Res.* 75 (2022), 747–793. https://doi.org/10.1613/jair.1.13685

[16] N. Sturtevant. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games* 4, 2 (2012), 144 – 148. http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf

[17] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems* 10, 05 (2002), 557–570.

[18] Hideaki Takahashi and Alex Fukunaga. 2024. Supplementary Material for "On the Transit Obfuscation Problem. *arXiv preprint* (2024).

[19] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. 2015. A survey on security and privacy issues in big data. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 202–207.

[20] Stanley Wasserman and Katherine Faust. 1994. Social network analysis: Methods and applications. (1994).