Accurate, Secure, and Efficient Semi-Constrained Navigation with Multiple Spatial Restrictions

***Abstract*—Graph-Based Service (GBS) facilitates navigation in multiple scenarios. Although convenient, it requires users to upload their locations to an untrusted navigation server. Such interaction violates location privacy of users who prioritize sensitive locations. Unfortunately, existing work neglects partial visit orders and geographic features, which motivates us to consider *partially ordered stops* and *circuitous path between two stops*. In this work, we advance the state-of-the-art on Secure Semi-Constrained Navigation by allowing Multiple Spatial Restrictions (SCN-MSR), while improving navigation accuracy without sacrificing security. Based on the Divide-and-Conquer strategy, we design four novel navigation algorithms NLow, NMid, NHigh, and secure skimming. They enable an untrustworthy navigation server to search over an encrypted dataset to find the shortest path via k stops, subject to following orders of several stops and circumventing road obstacles. Next, we provide formal security analysis under the semi-honest adversary model. Experimental results demonstrate that Gaia achieves high geometric alignment with theoretical optimal paths and reduces navigation time by approximately 60% compared to the state-of-the-art Hermes, requiring only about 1.6 s to search over 2,200 road intersections with 5 stops and 3 partial orders.**

***Index Terms*—Navigation, Unordered Stops, Spatial Restrictions, Security, Accuracy, Efficiency.**

I. Introduction

A. Background

Graph-Based Service (GBS) has flourished in recent years, facilitating graph-structured services, such as social search [1], communication network search [2] online web advertising [3], and navigation services [4]. Among different types of GBS, Shortest Path Computation (SPC) is crucial to both daily life and exceptional circumstances. For example, Google Maps and Waze are two of the most widely used navigation tools today, which enable users to get directions from a starting point S to a destination D. Meanwhile, military supply transport, disaster relief shipping, and critical medicine delivery also demand for SPC to improve efficiency.

In SPC, navigation users have to upload their real-time locations or entered locations to a Navigation Server (NS), which is computationally powerful. But NS is proven and assumed to be not trusted [5]–[12]. Such a common assumption further highlights the importance of location privacy since the leakage of sensitive location information can result in advertising harassment, social connection exposure, and persistent surveillance [13].

B. Existing Work

To address the location privacy leakage, Meng et al. [9] proposed a graph encryption scheme GRECS for approximate shortest distance queries. Following this work, there are several well-studied graph encryption-based schemes, such as graph encryption for shortest distance queries (SecGDB) [14], graph encryption for shortest path queries (SPQ) [15], privacy-preserving shortest distance queries (GENOA) [19], large graphs for secure queries (GraphShield) [16], and privacy-enhancing and accurate shortest path retrieval (CryptGraph) [17], PathGES [18]. Their key idea is to query a shortest distance or path in an encrypted graph while protecting location privacy.

In this work, we focus on unraveling new problems on securely navigating via k unordered stops based on the seminal paper in this line of research, i.e., the state-of-the-art (sota) Semi-Constrained Navigation (SCN) scheme Hermes [20]. Its main idea is to preprocess and encrypt the road graph and send a query token of k stops to the NS, which computes the shortest path over location ciphertexts. Especially, Hermes designs eight secure navigation algorithms by utilizing cryptographic techniques. Its technical crux is to leverage a **Divide-and-Conquer (DaC) navigation approach** to divide the service area into three main areas $MA^{\mathrm{L}} = \{A_1, A_2\}$, $MA^{\mathrm{M}} = \{A_3, A_4\}$, and $MA^{\mathrm{H}} = \{A_5, A_6\}$, then recursively invoke seven navigation algorithms to compute the next stop.

C. Problems and Motivations

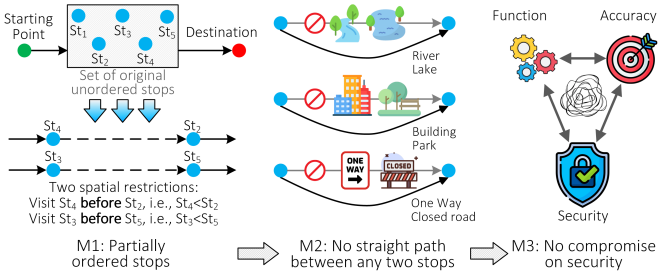

Fig. 1: Three Motivations behind This Work.

Now we discuss the Problems of existing work and present our Motivations while visualizing them in Fig. 1.

P1: Negligence of partial visit orders. The SCN fails to account for one actual demand that navigation users have Multiple Spatial Restrictions (MSRs) on the visiting order of two stops among k unordered stops. **M1 (Function-wise): Partially ordered stops**. For example, Alice has a set of stops $\{St_1, St_2, St_3, St_4, St_5\}$ and she expects to visit St_4 before St_2 and visit St_3 before St_5. We use $St_4 < St_2$, $St_3 < St_5$ to denote the two spatial restrictions. We use Preceding Stop

(PSt) and Succeeding Stop (SSt) to denote St_4 (St_3) and St_2 (St_5), respectively.

P2: Negligence of geographic features. First, the core design philosophy of Hermes' navigation strategy is assuming there is a straight path between two stops, thus not considering some real-world navigation obstacles between two locations on a map, e.g., river, lake, building, park, road closure, and one-way road. Second, the Vertically Nearest Hop (VNH) strategy from the DaC solution [20] is not entirely correct because it computes the first Bridge Point (BP) in MA^{M} from the perspective of the starting point, rather than the last visited stop in MA^{L}. **M2 (Accuracy-wise): Circuitous path between two stops**. First, we embrace the fact that that there is no straight path between two locations since there are different types of obstacles. We use $St_1 \nleftrightarrow St_2$ to denote a road closure between St_1 and St_2. We use current stop (CSt) and next stop (NSt) to denote St_1 and St_2, respectively. Second, we retain the geographic blocks between any two intersections (stops) and accept the impact of such features on the navigation.

M3 (Security-wise): No compromise on security. When realizing a new function and improving navigation accuracy, our new endeavour cannot sacrifice security. First, neither the navigation query nor the route computation can leak the partial order to the NSP. Second, both the navigation query and the route computation must not reveal to the NSP any partial information about the block between any two stops.

For the simple version of M1, i.e., only one spatial restriction $St_4 < St_2$ is considered, we have two potential solutions: First, the user can compute the combination of different routes based on the remaining unordered stops, submit corresponding navigation queries to the cloud, concatenate the navigation sub-paths, and compare the path distance of different routes. As in the example above, Alice has a potential route $S-St_4-St_2-D$ and she can insert St_1 and St_3 into the three slots, leading to nine queries. In other words, one spatial restriction on two stops from k stops will lead to $(2+1)^{k-1}$ queries, which is quite cumbersome and unnecessary. Another solution is to set the subsequently visited stop as a temporary destination and then march toward the original destination. This will get us two queries $(S, St_2; St_4)$ and $(St_2, D; \emptyset)$ in the example above. However, the same issue persists since we are not certain how to fit St_1 and St_3 in the navigation, which traps us in the traversal predicament.

For M2, the most intuitive solution is to start from CN, bypass the block, and land on the NSt. This can be done by navigating to one of the (at least) two directions, i.e., left front and right front. Meanwhile, the navigation does not end at NSt when there are still more unvisited stops. Blindly bypassing the block without considering remaining nodes will probably reduce navigation accuracy.

D. Our Approach and Technical Challenges

We propose *Secure Semi-Constrained Navigation with Multiple Spatial Restrictions, named* Gaia, to fulfill all the motivations. (1) We preprocess and encrypt an original road graph with CKKS [21]. (2) We observe the locations of PSt and SSt. For instance, we determine which area(s) St_2 and St_4 sit in, i.e., MA_{St_2} and MA_{St_4}, based on the distribution results of grouping strategy from DaC. If $MA_{St_4} < MA_{St_2}$, we invoke DaC. (3) During the preprocessing, we set a block indicator between any two nodes in the original dataset and plant a Deviation Stop (DSt) on the shortest path of any two straight-path-blocked stops to bypass the block between them.

TC1: How to achieve accurate, secure, and efficient through $k-2$ unordered stops and 2 ordered stops (PSt, SSt) when $MA_{PSt} = MA_{SSt}$? The original design of DaC did not consider partially ordered nodes, which not only created challenges for addressing the issue but also presented opportunities for algorithmic improvement. As portrayed in Fig. 2, when St_4's area is higher than St_2's, i.e., $MA_{St_4} << MA_{St_2}$, the navigation is unaffected. This holds true for case 1, case 3, and case 5 in Fig. 3 where $MA_{St_4} = MA_{St_2}$ as well. But in the other three cases in Fig. 3. These situations become more complex for four reasons. First, hopping to PSt before SSt in such situations conflicts with the existing navigation strategy. Second, there will be other unvisited stops being associated with accurately navigating in current area. Last, given that navigation in the plaintext domain already presents significant challenges, navigation in the ciphertext domain while correctly invoking navigation strategies of DaC, making the first TC more challenging than SCN.

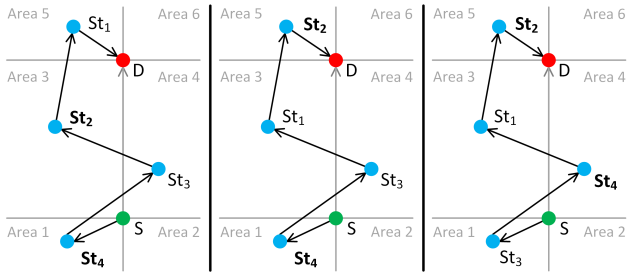

Fig. 2: Three Navigation Cases Unaffected by the Partial Visit Order ($St_4 < St_2$) when $MA_{St_4} << MA_{St_2}$.

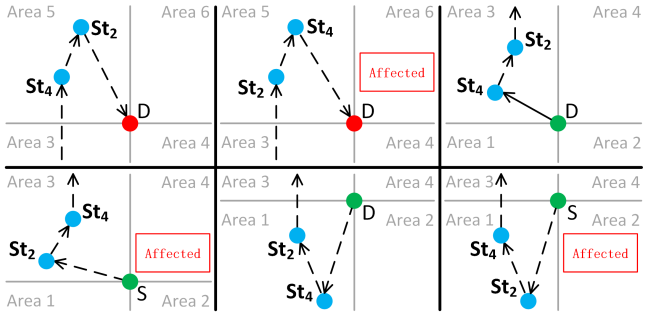

Fig. 3: Six Navigation Cases when $MA_{St_4} = MA_{St_2}$, Three of Which are Affected by the Partial Visit Order ($St_4 < St_2$).

TC2: How to achieve accurate, secure, and efficient SCN-MSR through $k-2$ unordered stops and 2 ordered stops (PSt, SSt) when $MA_{PSt} >> MA_{SSt}$? As illustrated in Fig. 4, MA_{PSt} is higher than MA_{SSt}. This situation resembles the one in

TC1 regarding hopping to PSt prior to SSt, conflicting with area visit priorities. The unique location of S implies that navigation will be trickier when other stops are situated in the two areas.

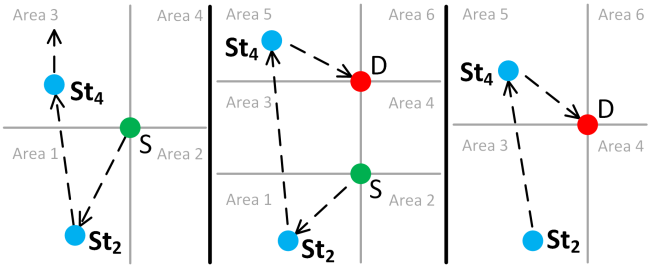

Fig. 4: Three Navigation Cases Affected by the Partial Visit Order ($St_4 < St_2$) when $MA_{St_4} >> MA_{St_2}$.

TC3: How to achieve accurate, secure, and efficient SCN-MSR through the planted DSs as an optional and partially ordered stop in different zones? As displayed in Fig. 5, due to the block between St_1 and St_2, we recruit a DSt as a detour indicator. But the DSt can situate in three areas, which call for different navigation strategies. For example, when standing in the unvisited zone above St_2, the DSt provides an optional way to hop forward. Nevertheless, the mutual correlations between DSt and the unvisited stops within this zone complicate the navigation. Hence, the DSs make TC3 more challenging than those encountered in prior work.

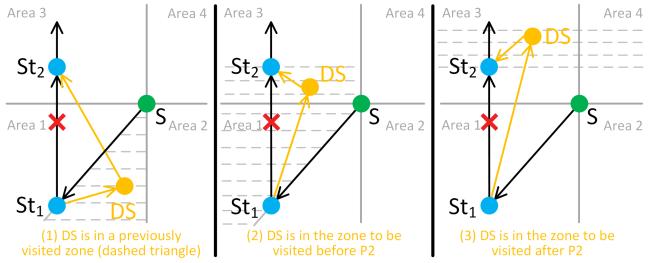

Fig. 5: Three Navigation Cases with $St_1 \nleftrightarrow St_2$ and A (yellow) Deviation Node between St_1 and St_2.

TC4: How to navigate from S into MA^{L} when we do not have a reference stop in MA^{M}? When the precomputed B_1 is abandoned for the purpose of improving accuracy, we lose a secondary navigation reference location besides S, disabling SS and SC. Such a predicament prevents us from initiating navigation to MA^{L}.

E. Our Contributions

For TC1, we formulate a strategy 1area-Wait (1W) when $A_{\mathbf{PSt}} = A_{\mathbf{SSt}}$. It checks whether the next hop is SSt before hopping to PSt. If not, the navigation continues; otherwise, it puts SSt in the set of visited stops in A_{PSt}, hops to the next hop until PSt is visited, then puts SSt back to the set of unvisited stops in A_{PSt}. Note that 1W strategy applies to three cases where both PSt and SSt are in MA^{H}, MA^{M}, and MA^{L}, respectively. **For TC2, we develop a strategy 2area-Delay (2D) when $A_{\mathbf{PSt}} >> A_{\mathbf{SSt}}$.** It puts SSt on hold, continues to hop to PSt, then circles back to SSt. **For TC3, we devise a strategy ViaDSt by observing the DSt' location when there is a DSt between the current stop (CSt) and the next stop (NSt).** If it is in a zone that is visited (to be visited before the next stop), there are certainly no other stops in the zone, we hop backward (forward) to DSt, then to NSt. Last, if it is in a zone to be visited after the next stop, we hop forward to DSt, then to NSt if there are no other stops. Otherwise, we draw an auxiliary line from CSt to NSt to observe their locations and decide which one to hop to. If the navigation advances clockwise, we hop to the leftmost stop located on the left (right) of $L^{\mathrm{NSt}}_{\mathrm{CSt}}$. If the navigation advances anticlockwise, we proceed similarly by changing the leftmost to the rightmost.

For TC4, we plan a strategy C-AC (Clockwise-AntiClockwise), i.e., navigating by observing the presence of stops in six subareas. For example, If A_3 has stops while A_4 does not, we navigate into A_2 first before A_1. Next, we design a new navigation algorithm Secure sKim (SK) to orient St_j against $\mathbf{BSt_i}$ at most n^{semi} times in a semispace of A^{L}.

Our contributions are summarized as follows.

- **New navigation function for Secure SCN.** We have realized a new navigation function, which allows users to impose multiple spatial restrictions on the k stops.
- **Improved navigation accuracy for Secure SCN.** We have designed new navigation strategies to ensure **geometric alignment** with the physical shortest path, significantly improving the navigation accuracy over ciphertexts.
- **Formal analysis and evaluation.** We have provided formal security analysis, complexity analysis, and implementations with extensive experiments.

Paper Organization. We review related work in Section II. We introduce system model, threat model, and design objectives in Section III. We revisit preliminaries in Section IV. In Section V, we present Gaia followed by formally security analysis in Section VI and performance analysis in Section VII. Finally, we conclude with Section VIII.

II. Related Work

To facilitate secure search over structured data, Chase and Kamara [22] pioneered the notion of Structured Encryption (STE). Meng et al. [9] designed GRECS including three oracle encryption schemes for approximate shortest distance queries by employing Homomorphic Encryption (HE), Pseudo-Random Function (PRF), and Pseudo-random Permutation (PRP). graph encryption for shortest distance queries (SecGDB) [14] Liu et al. [19] proposed GENOA, a 2-hop cover labeling [23] based graph encryption scheme to enable shortest distance queries on an encrypted and remotely stored graph. Ghosh et al. [15] presented the first graph encryption scheme GKT for Single-Pair Shortest Path (SPSP) queries based on a dictionary encryption scheme. Du et al. [16] answered shortest distance queries, other classic graph-based queries (e.g., maximum flow) and more complicated analytics

TABLE I: Comparison with Existing Schemes

Schemes	Tow-Cloud Model	Cryptographic Techniques	Location privacy		Unordered Stops	Spatial Restrictions	Accuracy	Search Efficiency
			Source	Destination				
GRECS [9]	×	HE, PRF, PRP	●	●	×	×	Approximate	
SecGDB [14]	√	HE, PRF, PRP, GC, OT	●	●	×	×	High	
GENOA [19]	×	HE, PRF	●	◐	×	×	Approximate	
GKT [15]	×	STE, PRF	●	●	×	×	High	
GraphShield [16]	√	HE, PRF, PRP, GC, OT	●	●	×	×	High	
CryptGraph [17]	√	HE, SeS	◐	◐	×	×	High	
PathGES [18]	√	EMM	●	●	×	×	High	
Hermes [20]	√	HE, PRF, PRP, GC, OT	●	●	×	×	Medium	
Gaia	√	HE, PRF, PRP, GC, OT	●	●	√	√	High	

(e.g., PageRank) by designing GraphShield, a structured encryption scheme for graphs by using HE, PRF, PRP, Garbled Circuits (GC), and Oblivious Transfer (OT). CryptGraph [17] designed CryptGraph to achieve efficient and accurate retrieval over encrypted graphs based on HE and Secret Sharing (SeS). It leveraged two non-colluding servers to collaboratively retrieve the shortest path and shortest distance. Although graph nodes, S, and D are permuted, still they are processed in plaintext. PathGES [18] is an improvement over GKT [15]. Based on response-hiding Encrypted MultiMap (EMM), it designs a new data structure that mitigates previous attacks while achieving non-interactive computation and fast query time. Hermes [20] introduced the notion of Semi-Constrained Navigation (SCN) and solved the secure SCN by putting forth the Divide-and-Conquer (DaC) navigation approach together with eight meticulously tailored navigation algorithms. We will provide a detailed introduction to DaC in Section IV.

These schemes have slightly different security goals or underlying techniques. Our work aims to achieve accurate, secure, and efficient navigation strategies and algorithms for semi-constrained navigation with multiple spatial restrictions. A comparative summary is provided in Table I.

III. Problem Statement

A. System Model

The system model of Gaia is depicted in Fig. 6, consisting of user $\mathcal{U}$, navigation server $\mathcal{S}$, and a navigation proxy $\mathcal{P}$. We list the key notations in Table II.

In the Initialization phase, the user preprocess a road dataset $\mathcal{RD}$ to obtain a road graph $\mathcal{RG}$ and generates cryptographic parameters. In the Encryption phase, the user encrypts $\mathcal{RG}$ to obtain path-distance oracles of nodes and an encrypted road graph $\mathcal{EG}$. The user uploads $\mathcal{ERG}$ to the NS and shares a partial secret key sk_1 from the secret key set $\mathcal{SK} = \{sk_1, sk_2, sk_3, sk_4\}$ with the navigation proxy. In the Query phase, the user has a navigation query nq with k unordered stops $\{St_1, St_2, \cdots, St_k\}$, generates a query token t_{nq}, and submits t_{nq} to the server. In the Search phase, the server searches $\mathcal{EG}$ by using t_{nq}, computes an encrypted shortest path Esp_{nq} with the assistance of the proxy, and returns Esp_{nq} along with the encrypted shortest distance Edt_{nq} to the user. In the Decryption phase, the user decrypts the result to recover the shortest path sp and shortest distance dt in plaintext.

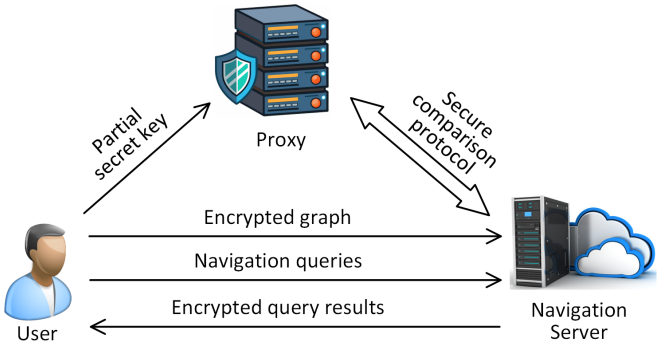

Fig. 6: System Model of Gaia.

TABLE II: Key Notations of TiMTAPS

Notation	Definition
SCN	Semi-Constrained Navigation
SCN-MSR	SCN with Multiple Spatial Restrictions
DaC, MA, A	Divide-and-Conquer, main area, area
S, D, St, BP	Starting point, Destination, Stop, Bridge Point
PSt, SSt	Preceding Stop, Succeeding Stop
CSt, NSt	Current Stop, Next Stop
HE	Homomorphic Encryption
PRF, PRP	Pseudo-Random Function/Permutation
GC, OT	Garbled Circuits, Oblivious Transfer
STE, SYE	Structured Encryption, Symmetric Encryption
$\mathcal{U}$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{RD}$	User, navigation server, proxy, road dataset
$\mathcal{RG}$, $\mathcal{ERG}$	Road dataset, road graph, encrypted road graph
nq, tk, PO	Navigation query, query token, partial order
$\mathcal{NQ}$, $\mathcal{ST}$	Set of navigation queries, set of stops
n, k	Number of nodes, number of stops
sp, dt, $\mathcal{L}$	Shortest path/disctance, leakage function
Σ, Ω,	DaC scheme, CKKS encryption scheme
F, P, H, h	PRF, PRP, random oracle, hash function
$\mathcal{SK}$, $\mathcal{PK}$	Secret key, public key
$\mathcal{ERD}$, $\mathcal{AR}$	Road graph dictionary, array
Esp, Edt	Encrypted shortest path/distance
$\mathcal{EQR}$	Encrypted query result
Nav, NLow	Overall algorithm, sub-algorithm for A^{L}
NMid, NHigh	Sub-algorithm for A^{M} and A^{H}
C, AC	Clockwise, anticlockwise
SK, **PP**	Secure sKimming, perpendicular vector

We formally define the SCN-MPR and the Secure SCN-MPR as follows.

Definition 1 (SCN-MPR). Given a starting point S, a destination D, a set of k stops $\mathcal{ST} = \{St_1, \cdots, St_k\}$, a set of t partial orders $\mathcal{PO} = \{O_1, \cdots, O_t\}$, and a navigation

function $\mathcal{NF}$, SCN-MPR is to find a sequence S of k different stops in $\mathcal{P}$ constituting a shortest path from S to D passing $\mathcal{ST}$:

$$\underset{St_{i_1},\cdots,St_{i_k}}{\arg\min}\ \mathcal{NF}(S,D,\mathcal{ST},\mathcal{PO}) = \{(St_{i_1},\cdots,St_{i_k}) \mid \forall \mathsf{S}(St_{j_1},\cdots,St_{k_k}) : \big(\mathsf{Dist}(S,D,St_{i_1},\cdots,St_{i_k}) \le \mathsf{Dist}(S,D,St_1,\cdots,St_k)\big) \wedge \big(St_{i1} < St_{i2},\ i \in [t]\big)\}.$$

Definition 2 (Secure SCN-MPR). A secure SCN-MPR scheme consists of the following five Probabilistic Polynomial-Time (PPT) algorithms:

- $(\mathcal{SK},\mathcal{PK},\mathcal{RG},\mathsf{SCGC}) \leftarrow \mathsf{Ini}(1^\lambda,\mathcal{RD})$: is a probabilistic generation algorithm executed by user. It takes security parameter λ and an road dataset $\mathcal{RD}$ as input, and outputs a secret key set $\mathcal{SK}$, a public key set $\mathcal{PK}$, a road graph $\mathcal{RG}$, and a secure comparison garbled circuit SCGC.
- $\mathcal{ERG} \leftarrow \mathsf{Enc}(\mathcal{SK},pk,\mathcal{RG})$: is a probabilistic algorithm executed by the user. It takes as input a secret key set $\mathcal{SK}$, a public key pk (part of $\mathcal{PK}$), and a road graph $\mathcal{RG}$, and outputs an encrypted road graph $\mathcal{ERG}$.
- $tk \leftarrow \mathsf{Query}(\mathcal{SK},nq)$: is a probabilistic algorithm executed by the user. It takes as input a secret key set $\mathcal{SK}$ and a navigation query nq, and outputs a query token $tq = (S,D,\mathcal{ST} = \{St_i\}_{i=1}^k, \mathcal{PO} = \{PO_i\}_{i=1}^t)$.
- $(\mathcal{EQR}) \leftarrow \mathsf{Search}(\mathcal{SK},\mathcal{PK},tk,\mathcal{ERG})$: is an interactive protocol executed among the user, the server, and the proxy. It takes a secret key set $\mathcal{SK}$, a public key set $\mathcal{PK}$, a query token tk, and an encrypted road graph $\mathcal{ERG}$ as input, outputs an encrypted result $\mathcal{EQR} = (Esp, Edt)$.
- $\mathcal{QR} \leftarrow \mathsf{Dec}(\mathcal{SK},\mathcal{EQR})$: is a deterministic algorithm executed by the user. It takes a secret key set $\mathcal{SK}$ and an encrypted query result $\mathcal{EQR}$, and outputs a query result $\mathcal{QR} = (sp, dt)$.

B. Security

Now we provide the security requirements for the secure SCN-MSR scheme. The user is honest and the proxy is trusted [16] From a high-level view, the security we expect from secure SCN-MSR scheme is that: (1) given $\mathcal{ERG}$, no adversary $\mathcal{A}$ can learn any information about $\mathcal{RG}$ and (2) given a polynomial number of Query executions for a set of SCN-MSR queries $\mathcal{NQ} = \{nq_1,\cdots,nq_w\}$, no adversary $\mathcal{A}$ can learn any partial information about nq_i or $\mathcal{RG}$.

Following [9], [15], [20], [24], [25], we parameterize the security definition with *leakage functions* for three phases of Gaia, namely Initialization, Query, and Search. Meanwhile, we also adopt the notion of *adaptive semantic security* of their works in Gaia.

Definition 3 (Adaptive Semantic Security). Let $\Pi = (\mathsf{Init},\mathsf{Enc},\mathsf{Query},\mathsf{Search},\mathsf{Dec})$ be a SCN-MSR scheme. We define two probabilistic experiments with a semi-honest adversary $\mathcal{A}$, a challenger $\mathcal{C}$, a simulator $\mathcal{S}$, and three leakage functions $\mathcal{L}_{\mathsf{Init}}$, $\mathcal{L}_{\mathsf{Query}}$, and $\mathcal{L}_{\mathsf{Search}}$:

Real$_{\mathcal{A}}(1^\lambda)$:
- $\mathcal{A}$ generates a road graph $\mathcal{RG}$.
- $\mathcal{C}$ calculates $(\mathcal{SK},\mathcal{PK},\mathcal{RG}) \leftarrow \mathsf{Ini}(1^\lambda,\mathcal{RD})$ and sends $\mathcal{ERG}$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a set of adaptively chosen SCN-MSR queries $\mathcal{NQ} = (nq_1,\cdots,nq_w)$. For a navigation query $nq_i,\ i \in [w]$, $\mathcal{A}$ receives $tk_i \leftarrow \mathsf{Query}(\mathcal{SK},nq_i)$ from $\mathcal{C}$, then $\mathcal{A}$ and $\mathcal{C}$ run $\mathsf{Search}(\mathcal{SK},\mathcal{PK},\ tk_i,\mathcal{ERG})$.
- $\mathcal{A}$ computes a bit b to be output by the experiment.

IDeal$_{\mathcal{A},\mathcal{S}}(1^\lambda)$:
- $\mathcal{A}$ outputs a road graph $\mathcal{RG}$.
- Given $\mathcal{L}_{\mathsf{Init}}$, $\mathcal{S}$ generate and sends $\mathcal{ERG}$ to $\mathcal{A}$.
- $\mathcal{A}$ produces a set of adaptively chosen SCN-MSR queries $\mathcal{NQ} = (nq_1,\cdots,nq_w)$. For each $nq_i,\ i \in [w])$, $\mathcal{S}$ is given $\mathcal{L}_{\mathsf{Query}}$ and $\mathcal{L}_{\mathsf{Search}}$. $\mathcal{A}$ and $\mathcal{S}$ simulate Search where $\mathcal{S}$ acts as a user (proxy) and $\mathcal{A}$ acts as the server.
- $\mathcal{A}$ computes a bit b to be output by the experiment.

We say that Π is adaptively $(\mathcal{L}_{\mathsf{Init}},\mathcal{L}_{\mathsf{Query}},\mathcal{L}_{\mathsf{Search}})$- semantically secure if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that

$$|\Pr[\mathbf{Real}_{\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda) = 1]| \le \mathrm{negl}(\lambda).$$

Definition 3 conveys that, given $\mathcal{ERG}$ and a view of the navigation protocol, $\mathcal{A}$ cannot acquire any information about $\mathcal{RG}$ beyond the leakage.

C. Leakage

Initiation leakage. The Initiation leakage includes the number of nodes (in $\mathcal{RG}$) n, the maximum distance Mdt, and the number of deviation nodes n_{dev}. **Query pattern leakage.** The query pattern QP indicates whether the points in a navigation query have been queried before. For two navigation queries nq_1 and nq_2, we define $\mathsf{Sim}(nq_1,nq_2) = (S_1 = S_2, D_1 = D_2, St_{11} = St_{21},\cdots,\ St_{1k} = St_{2k})$, i.e., whether the starting point, destination, and k stops in nq_1 match the ones in nq_2. Let $\mathcal{NQ} = (nq_1,\cdots,nq_w)$ be a set of navigation queries. The query pattern leakage function $\mathcal{L}_{\mathrm{QP}}(\mathcal{RG},nq_i)$ outputs an $m \times m$ matrix with entry $(i;j)$ being $\mathsf{Sim}(nq_i,nq_j)$. **Access pattern leakage.** The access pattern AP is the set of indices pointing to files that match the query. The access pattern leakage function $\mathcal{L}_{\mathrm{AP}}(\mathcal{RG},nq)$ for a road graph $\mathcal{RG}$ and a navigation query nq outputs the set of vertex identifiers that are retrieved for responding to nq.

D. Function, Accuracy and Efficiency

We also pay attention to both accuracy and efficiency. **Accuracy**. Given the stop visiting sequence, the SCN-MSR system should achieve high accuracy, regarding a low distance deviation rate and a high path similarity [20]. Such computation requires ground truth and equations, which we will introduce in Section VII. **Efficiency.** Despite the computation over ciphertexts, the SCN-MSR system should achieve acceptable efficiency regarding computational costs and communication overhead of entities as well as scalability with n and k.

IV. PRELIMINARIES

A. Homomorphic Encryption

The homomorphic encryption scheme we use in this work is the Cheon-Kim-Kim-Song (CKKS) scheme. It consists of five main algorithms:

$\mathsf{HGen}(\lambda)$: given a security parameter λ, chooses a power-of-two $M = M(\lambda, q_L)$, an integer $h = h(\lambda, q_L)$, an integer $P = P(\lambda, q_L)$, and a real value $\sigma = \sigma(\lambda, q_L)$; samples $s \leftarrow \mathsf{HWT}(h)$, $a \leftarrow \mathcal{R}_{q_L}$, and $e \leftarrow \mathcal{DG}(\sigma^2)$; samples $a' \leftarrow \mathcal{R}_{P \cdot q_L}$ and $e' \leftarrow \mathcal{DG}(\sigma^2)$; outputs a secret key $sk = (1, s)$, a public key $pk = (b, a) \in \mathcal{R}^2_{q_L}$ where $b = -as + e$, and an evaluation key $evk = (b', a') \in \mathcal{R}^2_{P \cdot q_L}$ where $b' = -a's + e' + Ps^2 \bmod P \cdot q_L$.

$\mathsf{HE}(pk, m)$: given the public key pk and a message m, samples $v \leftarrow \mathcal{ZO}(0.5)$ and $e_0, e_1 \leftarrow \mathcal{DG}(\sigma^2)$, outputs a ciphertext $c = v \cdot pk + (m + e_0, e_1) \bmod q_L$.

$\mathsf{HD}(sk, \mathbf{c})$: given the secret key sk and a ciphertext $\mathbf{c} = (b, a)$, outputs a message $b + a \cdot s \bmod q_l$.

$\mathsf{HAdd}(\mathbf{c}_1, \mathbf{c}_2)$: given two ciphertexts $\mathbf{c}_1$ and $\mathbf{c}_2$, outputs a new ciphertext $c_{\text{add}} = \mathbf{c}_1 + \mathbf{c}_2$.

$\mathsf{HMul}(\mathbf{c}_1, \mathbf{c}_2, evk)$: given two ciphertexts $\mathbf{c}_1 = (b_1, a_1)$ and $\mathbf{c}_2 = (b_2, a_2) \in \mathcal{R}^2_{q_l}$, sets $d_0, d_1, d_2) = (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \bmod q_l$m outputs a new ciphertext $c_{\text{mul}} = (d_0, d_1) + \lfloor P^{-1} \cdot evk \rceil \bmod q_l$.

B. Pseudo-Random Function, Pseudo-Random Permutation

We say that a polynomial-time computable function $\mathcal{F} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^*$ is a PRF if its output cannot be distinguished from the output of a random function f by any PPT distinguisher $\mathcal{D}$, i.e., $|\Pr[\mathcal{D}^{\mathcal{F}_k(\cdot)}(1^n) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(1^n) = 1]| \leq \text{negl}(\lambda)$ [27]. Simply put, $\mathcal{D}$ can access an oracle $\mathcal{PO}$ being either $\mathcal{F}$ or f. $\mathcal{D}$ can query $\mathcal{PO}$ at any point i to get $\mathcal{PO}(i)$. For every $\mathcal{D}$ that receives a description of $\mathcal{F}$ outputs 1 with almost the same probability as when it receives a description of a random function. A PRF is assumed to be a PRP when it is bijective.

C. Garbled Circuits and Oblivious Transfer

Yao introduced GC [28], [29] for secure two-party computation. The first party C called circuit generator creates a circuit $\widetilde{C}$ with wires. For each wire w_i, C chooses two random garbled values $\widetilde{w}_i^0, \widetilde{w}_i^1$, where $\widetilde{w}_i^j$ is the garbled value of w_i's value j. For each gate g_i, C creates a garbled table $\widetilde{T}_i$ that allows to obtain only the garbled value of the corresponding g_i's output given a set of garbled values of g_i's inputs. C sends the garbled tables, i.e., $\widetilde{C}$, to the second party called circuit evaluator E. E obtains the garbled inputs $\widetilde{w}_i$ corresponding to inputs of both parties via oblivious transfer. It evaluates $\widetilde{C}$ on the garbled inputs to obtain the garbled outputs simply by evaluating the garbled circuit gate by gate, using $\widetilde{T}$.

OT_l^t, parallel 1-out-of-2 OT of t l-bit strings [30], is a two-party protocol executed between a chooser A and a sender B. For $i = 1, 2, \cdots, t$, B inputs a pair of l-bit strings s_i^0, s_i^1 and A inputs t chosen bits b_i. Upon the completion of the protocol, A acquires the chosen strings $s_i^{b_i}$ but not the unchosen strings $s_i^{1-b_i}$, and B learns nothing about b_i.

D. Divide-and-Conquer

The general idea of DaC [20] is to first divide the k stops into three areas (MA^{L}, MA^{M}, MA^{H}) based on the position of S and the position of D, determine two bridge points B_1 and B_2 to connect the navigation in three areas, invoke three navigation strategies for the three areas. After each hop, DaC re-groups the stops in current area and starts the recursion. Next, we briefly introduce the navigation strategies and algorithms.

Secure Grouping (SG). Given S and D, the NS calculates a vector **SD** and the Cosine Similarity (CoSi) of **SD, SSt** and **DSt, DSt**. Given two vectors of attributes **A** and **B** sharing an angle θ, their CoSi is defined as:

$$\mathsf{Sim}(\mathbf{A}, \mathbf{B}) = \mathsf{Cos}(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}. \tag{1}$$

To simplify the grouping on St_i, only the sign of the Numerator of CoSi is required. The NS computes a ciphertext:

$$\begin{aligned} \text{ENCS'}^{\mathbf{SD}}_{\mathbf{SSt_i}} &= [\![(x_{st_i} - x_s)(x_d - x_s) + (y_{st_i} - y_s)(y_d - y_s)]\!]_1 [\![r]\!]_1 \\ &= [\![(x_{st_i} - x_s)(x_d - x_s) + (y_{st_i} - y_s)(y_d - y_s) + r]\!]_1. \end{aligned} \tag{2}$$

Next, the NS performs comparison on $\text{NCS'}^{\mathbf{SD}}_{\mathbf{SSt_i}}$ ($\text{NCS'}^{\mathbf{DSt}}_{\mathbf{DSt_i}}$) and $[\![r]\!]$ with the proxy based on GC and OT to insert St_i into one of the three areas.

Secure Bridging (SB). After SG, the NS computes a bridge point B_1 as the stop that is vertically nearest to S in MA^{M}. This is done by leveraging the Euclidean distance and CoSi to check all the stops in MA^{M} and compute the difference between two projection of two vectors ($\mathbf{SSt_i}$, $\mathbf{SSt_j}$) on **SD**:

$$\begin{aligned} &\mathsf{Euc}(S, St_i)\mathsf{Sim}(\mathbf{SSt_i}, \mathbf{SD}) - \mathsf{Euc}(S, St_j)\mathsf{Sim}(\mathbf{SSt_j}, \mathbf{SD}) \\ &= \frac{(x_{st_i} - x_{st_j})(x_d - x_s) + (y_{st_i} - y_{st_j})(y_d - y_s)}{\sqrt{(x_d - x_s)^2 + (y_d - y_s)^2}} \end{aligned} \tag{3}$$

Next, the NS sets D as the second bridge point B_2.

Secure Orient-Split-sCan-Hop (OSCH) for MA^{L}. The strategy works for navigation in MA^{L}.

- **Secure Orienting (SO).** The NS orients B_1 and each stops in $\mathcal{ST}^{\text{L}}$ to see which side of SD they locate in.
- **Secure Splitting (SS).** After SO, the NS splits the stops in the first semispace of A^{L} into two parts $\mathcal{ST}_L^{\text{semi}}$, $\mathcal{ST}_R^{\text{semi}}$ based on the vector $\mathbf{B_1S}$.
- **Secure Scanning (SC)).** After SP, the NS scans the St with the maximum or minimum angle $\angle StB_1S$ when stops are on $\mathbf{SB_1}$'s right or left side.

Secure Vertically Nearest Hop (SVNH) for MA^{M}. The strategy works for navigation in MA^{M}. It is already introduced in the computation of B_1.

Secure Orient-Split-Pull-Hop (OSPH) for MA^{H}. The strategy works for navigation in MA^{H}.

- **SO** is the same to one in OSCH.
- **SS** is similar to the one in OSCH except the reference vector is now $\mathbf{B_2St'}$ where St' is the last visited stop in MA^{M}.
- **Secure Pulling (SP)** is the same to the one in OSCH.

We refer the interested reader to [20] for more details.

V. The Proposed Navigation Scheme Gaia

A. Overview

Our scheme Gaia Π consists of five phases: Initialization, Encryption, Query, Search, and Decryption. The technical crux is elaborated in Search. Although our navigation rationale follows the DaC, we revise its algorithms to support MSR and optimize them to improve navigation accuracy. Π is built upon six building blocks as follows: a DaC scheme $\Sigma = (\mathsf{SG}, \mathsf{SB}, \mathsf{SO}^{\mathsf{L}}, \mathsf{SP}^{\mathsf{L}}, \mathsf{SC}^{\mathsf{L}}, \mathsf{SVNH}, \mathsf{SO}^{\mathsf{H}}, \mathsf{SS}^{\mathsf{H}}, \mathsf{SP}^{\mathsf{H}})$, a CKKS encryption scheme $\Omega = (\mathsf{HGen}, \mathsf{HE}, \mathsf{HD}, \mathsf{HAdd}, \mathsf{HMul})$, a PRF $\mathsf{F} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{*}$, a PRP $\mathsf{P} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{\lambda}$, a random oracle $\mathsf{H} : \{0,1\}^{*} \to \{0,1\}^{*}$, and a collision-resistant hash function h modeled as a random function [9]. Since we replace the BGN encryption scheme [31] used in Σ [20], some algorithmic changes are necessary, e.g., $Esd = Esd * Esd'$ is now $Esd = Esd + Esd'$.

B. Initialization

$\mathsf{Ini}(1^{\lambda}, \mathcal{RD})$ is run to generate $\mathcal{SK} = (k_1, k_2, k_3, k_4, sk)$, $\mathcal{PK} = pk$, $\mathcal{RG} = (\mathcal{V}, \mathcal{E})$, and a secure comparison garbled circuit SCGC. Here, $\mathcal{V}$ is the set of node (intersections), $\mathcal{E}$ is the set of undirected edges, and $l_{v_i v_j}$ is the length for each edge $(v_i, v_j) \in \mathcal{E}$. The user shares sk with the proxy through a secure channel. SCGC is invoked by the comparison in Σ.

C. Enhanced Floyd-Warshall Preprocessing with Spatial Constraints

Our scheme extends the traditional Floyd-Warshall algorithm to support spatial constraints processing.During this phase, we implement comprehensive processing for both blocked edges and partial order constraints.

The blockage handling mechanism marks blocked edges during preprocessing and precomputes DSt through a three-layer strategy to ensure path connectivity. This approach guarantees viable solutions under various blockage scenarios while fully utilizing Floyd precomputation results to avoid unnecessary global searches and achieve intelligent candidate node screening. We present the **algorithm DStSearch** in **Algorithm 1**.The three-layer strategy is detailed as follows:

Layer 1: Direct DSt Search → Layer 2: Neighbor Path Query → Layer 3: History Backtracking

- **Layer 1**: Identifies intermediate nodes that can directly connect blocked edges $i \to j$, screening candidate nodes through vertical projection distance,let $\mathcal{U}$ denote the set of nodes that have been already used.
- **Layer 2**: If Layer 1 fails, queries paths through neighbors of the blocked edge's starting point, utilizing Floyd precomputation results to directly retrieve shortest paths.
- **Layer 3**: If both previous layers fail, backtracks to previous nodes in the path history $\mathcal{H}$ for re-planning.

To control computational overhead, the search depth for neighbor exploration in Layer 2 and history backtracking in Layer 3 is strictly regulated.

Algorithm 1 DStSearch

Require: $i \to j$, $\mathcal{U}$, $\mathcal{H}$
Ensure: *DSt*

1: $DSt \leftarrow \mathsf{SecureSkimming}(i, j, \mathcal{U})$
2: **if** $DSt \neq -1 \wedge \mathsf{Connected}(i, DSt, j)$ **then**
3: **return** *DSt*
4: **end if**
5: $\mathcal{C} \leftarrow \emptyset$; // Candidates list
6: **for** each $neighbor \in \mathsf{Adjacency}(i) \wedge neighbor \notin \mathcal{U}$ **do**
7: **if** $\mathsf{PathExists}(neighbor, j)$ **then**
8: $distance \leftarrow \mathsf{Dist}(i, neighbor) + \mathsf{Dist}(neighbor, j)$
9: $path \leftarrow \mathsf{Build}(i, neighbor, \mathsf{GetPath}(neighbor, j))$
10: $\mathcal{C}.\mathsf{add}(\mathsf{Candidate}(neighbor, distance, path))$
11: **end if**
12: **end for**
13: $\mathcal{C}.\mathsf{sortByDistance}()$; // Shortest first
14: **if** $\mathcal{C} \neq \emptyset$ **then**
15: **return** $\mathcal{C}[0].firstHop$;
16: **end if**// Both Layer 1 and Layer 2 failed, proceed to backtracking
17: **if** $|\mathcal{H}| \geq 2$ **then**
18: $prev \leftarrow \mathcal{H}[|\mathcal{H}| - 1]$; // Previous node in path history
19: $newU \leftarrow \mathcal{U} \cup \{i\}$
20: Repeat lines 1-16 with $(prev, j, newU, \mathcal{H})$
21: **end if**

D. Encryption

Given $\mathcal{RG}$, the user computes the shortest path sp and shortest distance sd for each pair of nodes via the Floyd-Warshall algorithm [32]. Meanwhile, the user records a deviation node $St^{\text{dev}}_{v_i v_j}$ for (v_i, v_j) if there is no straight path between v_i and v_j. Next, the user encrypts $\mathcal{RG}$ by encrypting the information above as follows. For each node v_i with a pair of coordinates (x, y), the user randomly chooses a secret k_{v_i} and a number r_{v_i}. For each pair of (sp, sd) of v_i, the user creates an array $\mathcal{AR}_{v_i}$ with the item in the initial location

$$\mathcal{AR}_{v_i}[0] = ((\mathsf{HE}(pk, x) || \mathsf{HE}(pk, y)) \oplus H(k_{v_1} || r_{v_1}), r_{v_1}). \quad (4)$$

The user computes a permutation of N nodes $(\mathsf{P}(sk_1, v_1), \cdots, \mathsf{P}(sk_1, v_n))$ to randomize the entry order of n nodes. For each location $\mathsf{P}(sk_1, v_j)$, the user chooses a random number r_{v_j} and stores at $\mathcal{AR}_{v_i}[\mathsf{P}(sk_1, v_j)]$:

$$\left(sp_{v_i v_j} \oplus \mathsf{F}(sk_3, h(r_{v_j})) || Esd_{v_i v_j} || \mathsf{E}(St^{\text{dev}}_{v_i v_j})) \oplus H(k_{v_i} || r_{v_j}), r_{v_j}\right),$$

where $Esd_{v_i v_j} = \Omega.\mathsf{HE}(pk, sd_{v_i v_j})$.

Then, the user creates a road graph dictionary $\mathcal{RGD}$ to record pairs $(\mathsf{P}(sk_2, v_i), add_{v_i} || k_{v_i} \oplus \mathsf{F}(sk_4, v_i))$ for all $v_i \in \mathcal{V}$, where add_{v_i} is the location of $\mathcal{AR}_{v_i}$. Finally, the user sends $\mathcal{ERG} = (\mathcal{RGD}, \mathcal{AR})$ to the server. We depict an example of storing road information in $\mathcal{ERG}$ in Fig. 7.

E. Query

The user has a navigation query:

$$nq = (S, D, St_{i1}, \cdots, St_{ik}, PO_{j1}, \cdots, PO_{jt}), \quad (5)$$

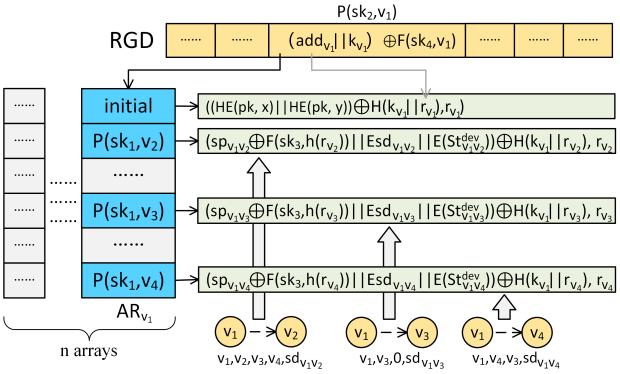

Fig. 7: An Example of Storing Road Information in $\mathcal{ERG}$.

where $PO_{jo} = (St^1_{jo}, St^2_{jo})$, $o \in [t]$ and runs $\mathsf{Query}(\mathcal{SK}, nq)$ to get a query token tk:

$$tk = (\mathsf{E}(S), \mathsf{E}(D), \mathsf{E}(St_{i1}), \cdots, \mathsf{E}(St_{tk})), \tag{6}$$

where $\mathsf{E}(S) = (\mathsf{P}(sk_2, S), \mathsf{F}(sk_4, S))$, $\mathsf{E}(D) = (\mathsf{P}(sk_2, D), \mathsf{F}(sk_4, D))$, $\mathsf{E}(St_{io}) = (\mathsf{P}(sk_2, St_{io}), \mathsf{F}(sk_4, St_{io}))$, $o \in [k]$.

Later, the user submits tk to the server and awaits for an encrypted query result.

F. Search

We give the overview of our navigation strategies in Fig. 8. The overall navigation algorithm Nav is shown in Alg. 2. It consists of four parts. First, it calls Secure Bridging (SB), i.e., $\Sigma.\mathsf{SG}(tk) = (\mathsf{E}(S), \mathsf{E}(D), \mathsf{E}(St_{i1}), \cdots, \mathsf{E}(St_{tk}))$, to group k stops into three sets. Next, it invokes three new sub-navigation algorithms, namely NLow, NMid, and NHigh, to determine the next stop in three main areas, respectively. In specific, NLow returns a stop that is used as a starting point for NMid, which does the same for H. Once the next hop is determined, the current algorithm recalls Nav, entering recursion.

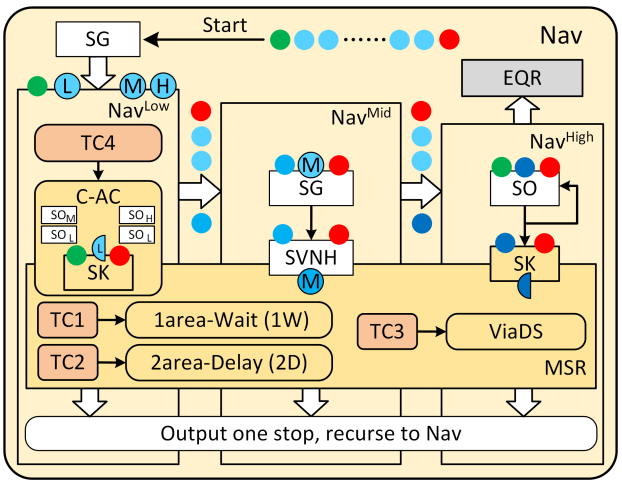

Fig. 8: Overview of Navigation Strategies.

We show the **algorithm NLow** in Algorithm 3. NLow executes our strategy C-AC (lines 3-35). In specific, it starts with observing the node existence $\mathsf{S}^{\mathrm{M}}_l$, $\mathsf{S}^{\mathrm{M}}_r$ of A^{M}'s two

Algorithm 2 Nav

Require: $\mathcal{ERG}$, tk
Ensure: $\mathcal{EQR}$
1: If $tk = \emptyset$: Return;
2: Parse $tk = (\mathsf{E}(S), \mathsf{E}(D), \mathsf{E}(St_{i1}), \cdots, \mathsf{E}(St_{tk}))$
3: Initialize $\mathcal{ST}^{\mathrm{L}}$, $\mathcal{ST}^{\mathrm{M}}$, $\mathcal{ST}^{\mathrm{H}}$, $\mathcal{ST}^{1\mathrm{W}}$, $\mathcal{ST}^{2\mathrm{D}}$, $\mathcal{EQR}$ as $\emptyset$;
4: $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}}) = \emptyset$, $\mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}}) = \emptyset$;//two last visited stops
5: $(\mathcal{ST}^{\mathrm{L}}, \mathcal{ST}^{\mathrm{M}}, \mathcal{ST}^{\mathrm{H}}) \leftarrow \Sigma.\mathsf{SG}(tk)$;//group the k stops
6: //Below, we omit $\mathcal{EQR}$, $\mathcal{EQR}$ for simplicity
7: $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}}) \leftarrow \mathsf{NLow}(\mathsf{E}(S), \mathsf{E}(D), \mathcal{ST}^{\mathrm{L}}, \mathcal{ST}^{\mathrm{M}}, \mathcal{ST}^{\mathrm{H}})$;
8: $\mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}}) \leftarrow \mathsf{NMid}(\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}}), \mathsf{E}(D), \mathcal{ST}^{\mathrm{M}})$;
9: $\mathsf{NHigh}(\mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}}), \mathsf{E}(S), \mathsf{E}(D), \mathcal{ST}^{\mathrm{H}})$;
10: Return $\mathcal{EQR}$;

semispaces (lines 4-10) and grouping stops in $\mathcal{ST}^{\mathrm{M}}$ into two sets $\mathcal{ST}^{\mathrm{L}}_l$, $\mathcal{ST}^{\mathrm{L}}_r$ (lines 11-17) by running $\Sigma.\mathsf{SO}$.

Next, NLow proceeds based on the four cases of $\mathsf{S}^{\mathrm{M}}_l$ and $\mathsf{S}^{\mathrm{M}}_r$, i.e., 10, 01, 11, and 00 (lines 18-30). **(1)** When $\mathsf{S}^{\mathrm{M}}_l = 1 \wedge \mathsf{S}^{\mathrm{M}}_r = 0$, we first hop from S to the right semispace of A^{M} and to the left semispace of A^{L} by invoking a new algorithm **Secure sKimming (SK)**. **(2)** For $\mathsf{S}^{\mathrm{M}}_l = 0 \wedge \mathsf{S}^{\mathrm{M}}_r = 1$, we hop to the left semispace of A^{M} and to the right semispace of A^{L}. **(3)** For $\mathsf{S}^{\mathrm{M}}_l = 1 \wedge \mathsf{S}^{\mathrm{M}}_r = 1$, we detect the node existence of A^{L}'s two semispaces, and SK with C and AC accordingly. **(4)** For $\mathsf{S}^{\mathrm{M}}_l = 0 \wedge \mathsf{S}^{\mathrm{M}}_r = 0$, we repeat the process by observing the the node existence of A^{H}'s two semispaces and A^{L}'s two semispaces. Last, NLow updates $\mathcal{EQR}$ and reignites Nav with current stop $\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}})$ and D (lines 33-35).

Before we dive into SK, we exemplify its navigation rationale with an example in Fig. 9. When there are stops only in Area 2, Area 3, and Area 4, we hop from S to the right semispace of A^{L}. To do so, we randomly select a St_1 from $\mathcal{ST}^{\mathrm{L}}$ and skim another stop St_2 in the remaining stops in this semispace to find the NorthEasternmost stop. We also need to consider two cases where the stops are on line L_1 perpendicular to **SD** and line SSt_1. After that, we check the MSRs before hopping to such a stop and reboot Nav.

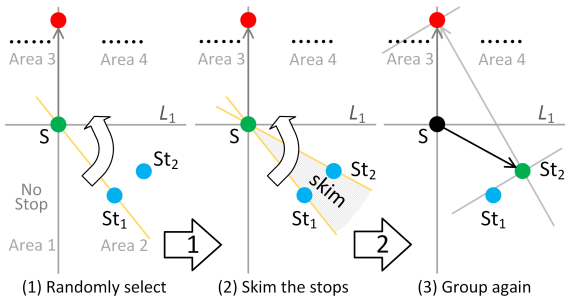

Fig. 9: An Example of Secure sKimming in A^{L}.

We show the **algorithm SK** in Algorithm 4. The algorithm is divided into two directions, i.e., C and AC. The **first direction** contains two parts. First, SK computes whether there are stops on line L_1 perpendicular to line SD (lines 4-9). If

Algorithm 3 NLow

Require: $\mathcal{ERG}$, $\mathsf{E}(S)$, $\mathsf{E}(D)$, $\mathcal{ST}^{\mathrm{L}}$, $\mathcal{ST}^{\mathrm{M}}$, $\mathcal{ST}^{\mathrm{H}}$, $\mathcal{EQR}$
Ensure: $\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}})$//it is $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}})$ when NLow is complete
1: **If** $\mathcal{ST}^{\mathrm{L}} == \emptyset$: return $\mathsf{E}(S)$;
2: **If** $|\mathcal{ST}^{\mathrm{L}}| == 1$: execute lines 42-43, return $\mathsf{E}(St) \in \mathcal{ST}^{\mathrm{L}}$;
3: Initialize $\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}}) = \emptyset$;
4: Initialize $\mathsf{S}^{\mathrm{M}}_l = 0$, $\mathsf{S}^{\mathrm{M}}_r = 0$;//two signs for A^{M}'s semispaces
5: Initialize IT as the first item from $\mathcal{ST}^{\mathrm{M}}$;
6: **while** IT $\neq \emptyset$ **do**//assign 0 or 1 to the two signs
7: **if** $\Sigma.\mathsf{SO}(\mathsf{E}(S), \mathsf{E}(D), \mathrm{IT}) == L$: $\mathsf{S}^{\mathrm{M}}_l = 1$;
8: **else** $\mathsf{S}^{\mathrm{M}}_r = 1$;
9: Set IT as the next item from $\mathcal{ST}^{\mathrm{M}}$;
10: **end while**
11: Initialize $\mathcal{ST}^{\mathrm{L}}_l = \emptyset$, $\mathcal{ST}^{\mathrm{L}}_r = \emptyset$;//$A^{\mathrm{L}}$'s left set and right set
12: Initialize IT as the first item from $\mathcal{ST}^{\mathrm{L}}$;
13: **while** IT $\neq \emptyset$ **do**
14: **if** $\Sigma.\mathsf{SO}(\mathsf{E}(S), \mathsf{E}(D), \mathrm{IT}) == L$: add IT into $\mathcal{ST}^{\mathrm{L}}_l$;
15: **else** add IT into $\mathcal{ST}^{\mathrm{L}}_r$;
16: Set IT as the next item from $\mathcal{ST}^{\mathrm{L}}$;
17: **end while**
18: **if** $\mathsf{S}^{\mathrm{M}}_l = 1 \wedge \mathsf{S}^{\mathrm{M}}_r = 0$ **then**//Low right, Low left
19: Retrieve the first item from $\mathcal{ST}^{\mathrm{L}}_r$ as IT;
20: $\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}}) = \mathsf{SK}(\mathsf{E}(S), \mathsf{E}(D), \mathrm{IT}, \mathcal{ST}^{\mathrm{L}}_r, C)$;//Clockwise
21: **else if** $\mathsf{S}^{\mathrm{M}}_l = 0 \wedge \mathsf{S}^{\mathrm{M}}_r = 1$ **then**//Low left, Low right
22: Repeat lines 19-20 with $\mathcal{ST}^{\mathrm{L}}_l$, AC;//AntiClockwise
23: **else if** $\mathsf{S}^{\mathrm{M}}_l = 1 \wedge \mathsf{S}^{\mathrm{M}}_r = 1$ **then**//look A^{L}
24: Repeat lines 4-10 with $\mathsf{S}^{\mathrm{L}}_l$, $\mathsf{S}^{\mathrm{L}}_r$, $\mathcal{ST}^{\mathrm{L}}$;
25: **if** $\mathsf{S}^{\mathrm{L}}_l = 1 \wedge \mathsf{S}^{\mathrm{L}}_r = 0$ **then**
26: Repeat lines 19-20 with $\mathcal{ST}^{\mathrm{L}}_l$, C;
27: **else if** $\mathsf{S}^{\mathrm{L}}_l = 0 \wedge \mathsf{S}^{\mathrm{L}}_r = 1$ **then**
28: Repeat lines 19-20 with $\mathcal{ST}^{\mathrm{L}}_r$, AC; //Low right, Clockwise. Excluded 00 in line 1
29: **else**: Repeat lines 19-20 with $\mathcal{ST}^{\mathrm{L}}_r$, C;
30: **end if**
31: **else**: Repeat lines 4-30 with $\mathsf{S}^{\mathrm{H}}_l$, $\mathsf{S}^{\mathrm{H}}_r$, $\mathcal{ST}^{\mathrm{H}}_l$, $\mathcal{ST}^{\mathrm{H}}_r$;
32: **end if**
33: $Esp \cup = \{(\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}}), sp_{SSt^{\mathrm{L}}_{1\mathrm{st}}} \oplus \mathsf{F}(sk_3, h(r_{St^{\mathrm{L}}_{1\mathrm{st}}})), r_{St^{\mathrm{L}}_{1\mathrm{st}}})\}$;
34: $Edt+ = Esd_{SSt^{\mathrm{L}}_{1\mathrm{st}}}$;
35: $\mathsf{Nav}(\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}}), \mathsf{E}(D), \mathcal{ST}^{\mathrm{L}}_r \setminus \{\mathsf{E}(St^{\mathrm{L}}_{1\mathrm{st}})\}))$;

$\mathcal{ST}^{\mathrm{On}}_{L_1}$ is not empty, we search the nearest stop to S on L_1 (lines 10-24). Before outputting the next stop $\mathsf{E}(St_1)$, we first check if it is in the waiting set $\mathcal{ST}^{\mathrm{1W}}$. If $\mathsf{E}(St_1)$ in waiting has a unvisited PSt, SK continues to skim in $\mathcal{ST}$ without $\{\mathsf{E}(St_1)\}$ or MSR; otherwise, SK deletes $\mathsf{E}(St_1)$ from $\mathcal{ST}^{\mathrm{1W}}$. If $\mathsf{E}(St_1)$ is not in $\mathcal{ST}^{\mathrm{1W}}$, we check by 2D. Then, SK checks the multiple spatial restrictions by invoking MSR and returns $\mathsf{E}(St_1)$. Second, if there are no stops on L_1, SK continues to skim stops in a clockwise manner and monitor whether SSt_1 has more than one stops (lines 30-36) or not (lines 38-41). The **second direction** resembles the first one, except that we replace R in line 26 with L to represent anticlockwise. For instance, when there are stops only in Area 1, Area 3, and Area 4, we hop from S to the left semispace of A^{L}. The other difference is that we look for the NorthWesternmost stop.

Algorithm 4 SK

Require: $\mathsf{E}(S)$, $\mathsf{E}(D)$, IT, $\mathcal{ST}$, DR, $\mathcal{PO}$
Ensure: $\mathsf{E}(St)$
1: **If** $|\mathcal{ST}| = 1$: Return $\mathsf{E}(St) \in \mathcal{ST}$;
2: Initialize $\mathsf{E}(St) = \emptyset$;
3: **if** DR == C **then**
4: Initialize $\mathsf{E}(St_i)$ as the first item from $\mathcal{ST}$;
5: Initialize $\mathsf{E}(St_1)$, $\mathsf{E}(St_2)$, and $\mathcal{ST}^{\mathrm{On}}_{L_1}$ as $\emptyset$;
6: **for** each $\mathsf{E}(St_i)$ in $\mathcal{ST}$//search stops on L_1
7: Compute $\mathrm{ENCS}^{\mathbf{DSt}}_{\mathbf{SSt_i}}$ and $\mathrm{ENCS}'^{\mathbf{DSt}}_{\mathbf{SSt_i}}$;
8: $cb \leftarrow \mathsf{SCGC}(r, \mathrm{NCS}'^{\mathbf{DSt}}_{\mathbf{SSt_i}})$;
9: **if** $cb = 1$: insert $\mathsf{E}(St_i)$ into $\mathcal{ST}^{\mathrm{On}}_{L_1}$;
10: **if** $\mathcal{ST}^{\mathrm{On}}_{L_1} \neq \emptyset$ **then**//search the nearest stop to S on L_1
11: Set $\mathsf{E}(St_1)$ as the first item of $\mathcal{ST}^{\mathrm{On}}_{L_1}$;
12: Set $\mathsf{E}(St_2)$ as the second item of $\mathcal{ST}^{\mathrm{On}}_{L_1}$;
13: **While** $\mathsf{E}(St_2) \neq \emptyset$
14: Compute $\mathrm{ENum}^{\mathbf{St_1},\mathbf{St_2}}_{\mathbf{PP_{SD}},S}$, $\mathrm{ENum}'^{\mathbf{St_1},\mathbf{St_2}}_{\mathbf{PP_{SD}},S}$;
15: $cb \leftarrow \mathsf{SCGC}(r, \mathrm{Num}'^{\mathbf{St_1},\mathbf{St_2}}_{\mathbf{PP_{SD}},S})$;
16: //$\mathbf{PP_{SD}}$ is a perpendicular vector of $\mathbf{SD}$
17: **if** $cb = 0$: $\mathsf{E}(St_1) = \mathsf{E}(St_2)$;
18: **if** $\mathsf{E}(St_1) \in \mathcal{ST}^{\mathrm{1W}}$
19: **if** $\mathsf{OPSt}(\mathsf{E}(St_1))$ is unvisited; //PSt of $\mathsf{E}(St_1)$
20: Invoke SK without $\{\mathsf{E}(St_1)\}$ or MSR;//1W
21: **else**: Delete $\mathsf{E}(St_1)$ from $\mathcal{ST}^{\mathrm{1W}}$;//Can visit St_1
22: **if** $\mathsf{E}(St_1) \in \mathcal{ST}^{\mathrm{2D}}$
23: Repeat lines 19-21 with $\mathcal{ST}^{\mathrm{2D}}$;//2D
24: **else** $\mathsf{E}(St_1) =$
25: $\mathsf{MSR}(\mathsf{E}(S), \mathsf{E}(St_1), \mathcal{ST}, \mathcal{PO}, \mathcal{ST}^{\mathrm{1W}}, \mathcal{ST}^{\mathrm{2D}})$
26: Return $\mathsf{E}(St_1)$;
27: **else** //no stop on the line L_1
28: Set $\mathsf{E}(St_2)$ as the second of $\mathcal{ST}$; $\mathcal{ST}^{\mathrm{On}}_{\mathbf{SSt_1}} = \emptyset$
29: **while** $\mathsf{E}(St_2) \neq \emptyset$ **do**//search the NEmost stop
30: **if** $\Sigma.\mathsf{SO}(\mathsf{E}(S), \mathsf{E}(St_1), \mathsf{E}(St_2)) = R$
31: $\mathsf{E}(St_1) = \mathsf{E}(St_2)$, Clear $\mathcal{ST}^{\mathrm{On}}_{\mathbf{SSt_1}}$
32: **else**//store the nearest stop to S on line $\mathbf{SSt_1}$
33: Compute $\mathrm{ENCS}^{\mathbf{PP_{SSt_1}}}_{\mathbf{SSt_2}}$ and $\mathrm{ENCS}'^{\mathbf{PP_{SSt_1}}}_{\mathbf{SSt_2}}$;
34: $cb \leftarrow \mathsf{SCGC}(r, \mathrm{NCS}'^{\mathbf{PP_{SSt_1}}}_{\mathbf{SSt_2}})$;
35: **if** $cb = 1$: insert $\mathsf{E}(St_2)$ into $\mathcal{ST}^{\mathrm{On}}_{\mathbf{SSt_1}}$;
36: Set $\mathsf{E}(St_2)$ as the next item of $\mathcal{ST}$;
37: **end while**
38: **if** $\mathcal{ST}^{\mathrm{On}}_{\mathbf{SSt_1}} \neq \emptyset$//search the nearest stop on $\mathbf{SSt_1}$
39: Repeat lines 11-26 with $\mathbf{PP_{SSt_1}}$;
40: **else** Repeat 25-26;//found one stop not on $\mathbf{SSt_1}$
41: **end if**
42: **else if** DR== AC **then**
43: Repeat lines 4-41 with L in line 30;
44: **end if**

Now we introduce the navigation rationale of MSR, which consists of three navigation strategies, i.e., 1area-Wait (1W), 2area-Delay (2D), and ViaDSt. The first two strategies correspond to the first two technical challenges and we give three examples in Fig. 10. For ViaDSt, we show two

examples when NSt and DSt are in the same main area, and we need to skim clockwise the leftmost stop from CSt Fig. 11.

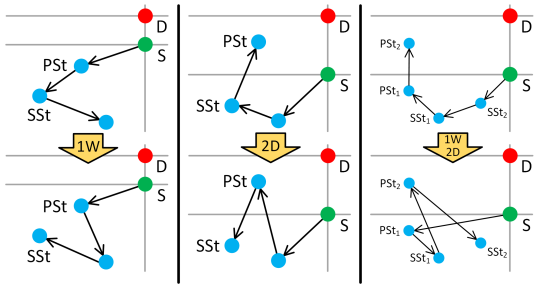

Fig. 10: Three Examples of Navigation Strategies (1W, 2D) for Handling Partial Orders.

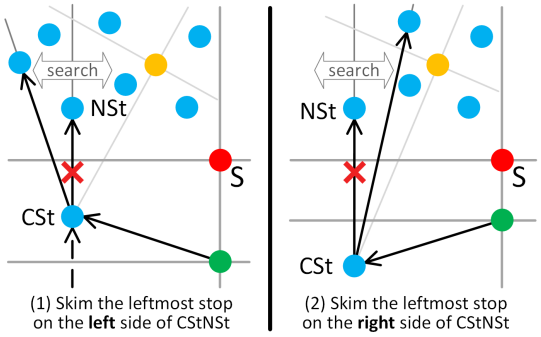

Fig. 11: Two Examples of Strategy ViaDSt when NSt and DSt are in the same main area.

We show the **algorithm MSR** in Algorithm 5. First, it checks whether the current stop St_1 to be hopped to is a SSt of anther stop and they are in the same main area (**1W**, lines 2-4). If so, MSR inserts St_1 into a waiting set $\mathcal{ST}^{1W}$ and skims through $\mathcal{ST} \setminus \{\mathsf{E}(St_1)\}$ without MSR. Second, if such a PSt is in a higher main area (**2D**, lines 5-7), MSR inserts St_1 into a delaying set $\mathcal{ST}^{2D}$ and skims similarly.

If there is a deviation node between $\mathsf{E}(S)$ (CSt) and $\mathsf{E}(St_1)$ (NSt), MSR applies the strategy **ViaDSt** (lines 8-16). When the deviation node DSt is in a visited main area, which has no unvisited stop, or it is in a main area that is lower than St_1, we MSR informs its caller to hop to DSt. If DSt and St_1 are in the same main area (lines 11-16), MSR computes the skimming direction and then skims from CSt.

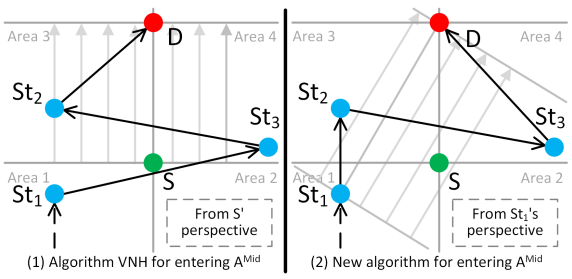

Fig. 12: An Example of Entering MA^{M}.

Algorithm 5 MSR

Require: $\mathsf{E}(S)$, $\mathsf{E}(D)$, $\mathsf{E}(St_1)$, $\mathcal{ST}$, $\mathcal{PO}$, $\mathcal{ST}^{1W}$, $\mathcal{ST}^{2D}$
Ensure: $\mathsf{E}(St)$

1: Initialize $\mathsf{E}(St) = \emptyset$
2: **if** $\exists\ i \in [t]$, $\big((PO_i[1] == \mathsf{E}(St_1)) \wedge (PO_i[0]$ is unvisited$)\wedge(MA_{PO_i[0]} == MA_{PO_i[1]})\big)$ **then**//St_1 is a SSt and it resides in same main area as PSt
3: Insert $\{\mathsf{E}(St_1)\}$ into $\mathcal{ST}^{1W}$;
4: $\mathsf{E}(St) \leftarrow$ SK in $\mathcal{ST} \setminus \{\mathsf{E}(St_1)\}$, MSR;//1W: output the second-best stop in $\mathcal{ST}$ omitting $\mathsf{E}(St_1)$
5: **else if** $\exists\ i \in [t]$, $\big((PO_i[1] == \mathsf{E}(St_1)) \wedge (PO_i[0]$ is unvisited$) \wedge(MA_{PO_i[0]} >> MA_{PO_i[1]})\big)$ **then**//St_1 is a SSt and it is in a lower main area than PSt
6: Insert $\{\mathsf{E}(St_1)\}$ into $\mathcal{ST}^{2D}$;
7: $\mathsf{E}(St) \leftarrow$ SK in $\mathcal{ST} \setminus \{\mathsf{E}(St_1)\}$, MSR;//2D: output the second-best stop in $\mathcal{ST}$ omitting $\mathsf{E}(St_1)$
8: **else if** $\exists\ \mathsf{E}(DSt)$ (NSt) between $\mathsf{E}(S)$ (CSt) and $\mathsf{E}(St_1)$ **then**//$\mathsf{E}(St_1)$ is neither in $\mathcal{ST}^{1W}$ nor in $\mathcal{ST}^{2D}$, ViaDSt:
9: **if** $MA_{\mathsf{E}(DSt)}$ is visited): Return $\mathsf{E}(DSt)$;
10: **if** $MA_{\mathsf{E}(DSt)} << MA_{\mathsf{E}(St_1)}$): Return $\mathsf{E}(DSt)$;
11: **else if** $MA_{\mathsf{E}(DSt)} == MA_{\mathsf{E}(St_1)}$
12: Call partial C-AC to get DR;
13: **if** DR $== L$
14: Invoke SK with C;
15: **else**: Repeat lines 14-15 with AC;
16: **end if**
17: Return $\mathsf{E}(St)$;

Now we explain the navigation rationale of NMid with Fig. 12. The left sub-figure shows the original VHN that navigates from S into A^{Mid}. However, the algorithm is flawed in its initial design because the navigation into A^{Mid} should be started from the last visited stop in A^{Low}. Such an observation dries us to come up with the right sub-figure, where we put forth the NMid to group via (St_1, D) and then invoke VNH. By doing so, we can improve the navigation accuracy.

We show the **algorithm NMid** in Algorithm 6. It first groups the remaining unvisited stops. If the low main area has unvisited stops, we invoke NMid to complete the navigation in A^{L}. Next, NMid invokes Σ.VNH to compute the next hop $\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}})$. Last, NMid recurses to Nav from $\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}})$ to $\mathsf{E}(D)$.

Finally, we come to the **algorithm NHigh** in Algorithm 7. It adopts strategy C-AC to observe the location of $St^{\mathrm{M}}_{\mathrm{last}}$ and node existence in A^{H}'s two semispaces (lines 3-10).

G. Decryption

After receiving $\mathcal{EQR} = (Esp, Edt)$, the user decrypts it as follows. For each triad $\gamma = (\mathcal{I}(St), sp \oplus \mathsf{F}(sk_3, h(r_{st})), r_{st})$ in Esp, user recovers sp by retrieving the r_{st_p} corresponding to $\mathsf{P}(sk_1, v_{st})$ and computing $sp = \gamma[1] \oplus \mathsf{F}(sk_3, h(\gamma[2])$; concatenate all recovered sp to form the complete navigation path $sp = S\text{-}St'_1\text{-}\cdots\text{-}D$. User decrypts Edt by computing $\mathsf{HD}(sk, Edt)$ to get the path distance dt.

Algorithm 6 NMid

Require: $\mathcal{ERG}$, $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}})$, $\mathsf{E}(D)$, $\mathsf{E}(S)$, $\mathcal{ST}^{\mathrm{Mid}}$, $\mathcal{EQR}$
Ensure: $\mathsf{E}St^{\mathrm{M}}_{1\mathrm{st}}$
1: **If** $\mathcal{ST}^{\mathrm{Mid}} == \emptyset$: Return $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}})$;
2: **If** $|\mathcal{ST}^{\mathrm{Mid}}| == 1$: execute line 8-9, return $\mathsf{E}(St) \in \mathcal{ST}^{\mathrm{Mid}}$;
3: Initialize $\mathsf{E}St^{\mathrm{M}}_{1\mathrm{st}}$, $\mathcal{ST}^{\mathrm{L}}$, $\mathcal{ST}^{\mathrm{M}}$, $\mathcal{ST}^{\mathrm{H}}$ as $\emptyset$;
4: $(\mathcal{ST}^{\mathrm{L}}, \mathcal{ST}^{\mathrm{M}}, \mathcal{ST}^{\mathrm{H}}) \leftarrow \Sigma.\mathsf{SG}(\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}}), \mathsf{E}(D), \mathcal{ST}^{\mathrm{Mid}})$;
5: **if** $\mathcal{ST}^{\mathrm{L}} \neq \emptyset$
6: Initialize $\mathsf{E}(St^{\mathrm{L}}_{\mathrm{last}}) \leftarrow \mathsf{NLow}(\cdots)$, go to line 5;
7: $\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}) \leftarrow \Sigma.\mathsf{VNH}(\mathsf{E}St^{\mathrm{L}}_{\mathrm{last}}, \mathsf{E}(D), \mathcal{ST}^{\mathrm{M}})$;
8: if $\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}) == \mathsf{E}(D)$: $\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}) = \mathsf{E}St^{\mathrm{L}}_{\mathrm{last}}$;//$\mathcal{ST}^{\mathrm{M}} == \emptyset$
9: $Esp\cup = \{(\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}), sp_{SSt^{\mathrm{M}}_{1\mathrm{st}}} \oplus \mathsf{F}(sk_3, h(r_{St^{\mathrm{M}}_{1\mathrm{st}}})), r_{St^{\mathrm{M}}_{1\mathrm{st}}})\}$;
10: $Edt+ = Esd_{SSt^{\mathrm{M}}_{1\mathrm{st}}}$;
11: $\mathsf{Nav}(\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}), \mathsf{E}(D), \mathcal{ST}^{\mathrm{Mid}} \setminus \{\mathsf{E}(St^{\mathrm{M}}_{1\mathrm{st}}\})$;

Algorithm 7 NHigh

Require: $\mathcal{ERG}$, $\mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}})$, $\mathsf{E}(S)$, $\mathsf{E}(D)$, $\mathcal{ST}^{\mathrm{H}}$, $\mathcal{EQR}$
Ensure: $\mathsf{E}St^{\mathrm{H}}_{1\mathrm{st}}$
1: **If** $\mathcal{ST}^{\mathrm{H}} == \emptyset$: Return $\mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}})$;
2: **If** $|\mathcal{ST}^{\mathrm{H}}| == 1$: execute lines 11-12, return $\mathsf{E}(St) \in \mathcal{ST}^{\mathrm{H}}$;
3: Repeat lines 4-10 of NLow with $\mathsf{S}^{\mathrm{H}}_l$, $\mathsf{S}^{\mathrm{H}}_R$;
4: **if** $\Sigma.\mathsf{SO}(\mathsf{E}(S), \mathsf{E}(D), \mathsf{E}(St^{\mathrm{M}}_{\mathrm{last}})) == L$ **then**
5: **if** $\mathsf{S}^{\mathrm{H}}_l = 1 \wedge \mathsf{S}^{\mathrm{H}}_r = 0$: Invoke SK with C;
6: **else if** $\mathsf{S}^{\mathrm{H}}_l = 0 \wedge \mathsf{S}^{\mathrm{H}}_r = 1$: Invoke SK with AC;
7: **else if** $\mathsf{S}^{\mathrm{H}}_l = 1 \wedge \mathsf{S}^{\mathrm{H}}_r = 1$: Invoke SK with C;
8: **else** $\mathsf{E}St^{\mathrm{H}}_{1\mathrm{st}} = \mathsf{E}(D)$;
9: **else** Repeat lines 5-7 with AC, C, AC;
10: **end if**
11: $Esp\cup = \{(\mathsf{E}(St^{\mathrm{H}}_{1\mathrm{st}}), sp_{St^{\mathrm{M}}_{\mathrm{last}}St^{\mathrm{H}}_{1\mathrm{st}}} \oplus \mathsf{F}(sk_3, h(r_{St^{\mathrm{H}}_{1\mathrm{st}}})), r_{St^{\mathrm{H}}_{1\mathrm{st}}})\}$;
12: $Edt+ = Esd_{St^{\mathrm{M}}_{\mathrm{last}}St^{\mathrm{H}}_{1\mathrm{st}}}$;
13: $\mathsf{Nav}(\mathsf{E}(St^{\mathrm{H}}_{1\mathrm{st}}), \mathsf{E}(D), \mathcal{ST}^{\mathrm{H}} \setminus \{\mathsf{E}(St^{\mathrm{H}}_{1\mathrm{st}}\})$;

VI. Security Analysis

Now we formally prove the security of Gaia.

Theorem 1. If $\mathcal{P}$ and F are pseudo-random, Ω is CPA-secure, then Π is adaptively $(\mathcal{L}_{\mathsf{Init}}, \mathcal{L}_{\mathsf{Query}}, \mathcal{L}_{\mathsf{Access}})$-semantically secure in the random oracle model where $\mathcal{L}_{\mathsf{Init}}(\mathcal{ERG}) = (n, Mdt, n_{dev})$.

Proof. We construct a simulator $\mathcal{S}$ working as below. Given $\mathcal{L}_{\mathsf{Init}}$, it samples:

$$\tau_{ji} \leftarrow \{0,1\}^{2|\mathsf{HE}|+l} \text{ for all } i \in [n], j = 0,$$
$$\tau_{ij} \leftarrow \{0,1\}^{\log n+2|\mathsf{HE}|+l} \text{ for all } 1 \le i \le n, 1 \le j \le n.$$

Next, it stores ζ_i in a $n+1$-element array $\mathcal{AR}_i$. For all $1 \le i \le n$, it samples $loc_i \leftarrow \{0,1\}^{\log n}$ without repetition and sets $\mathcal{RGD}[loc_i] \leftarrow \{0,1\}^{\log n+l}$.

Given $\mathcal{L}_{\mathsf{Query}}$ and $\mathcal{L}_{\mathsf{Access}}$, $\mathcal{S}$ checks whether any of S, D, and k stops $\{St_1, \cdots, St_k\}$ appeared in any previous navigation query. If S did, $\mathcal{S}$ sets $\mathsf{E}(S) = (\mathsf{E}(S)[0], \mathsf{E}(S)[1])$ to previous values. If not, it sets $\mathsf{E}(S)[0] = loc_i$ for a unoccupied $\mathsf{E}(S)[1]$ and as below. It randomly selects a previously unused $\alpha \in [n]$, a key $k_v \leftarrow \{0,1\}^l$, and sets $\mathsf{E}(S)[1] = \mathcal{EGD}[\mathsf{E}(S)[0]] \oplus < \alpha || k_v >$. It records the association between k_S and $\mathsf{E}(S)[0]$ and $\mathcal{ERD}[\mathsf{E}(S)[0]]$ It does the same for D and k stops analogously.

$\mathcal{S}$ simulates H as below. Given (k_v, r) as input, $\mathcal{S}$ checks whether: (1) k_v has been queried previously; (2) if any entry in $\mathcal{AR}$ has the form $< str, r >$ where str is a $(\log n + 2|\mathsf{HE}|)$-bit string. If k_v is queried previously, $\mathcal{S}$ creates a counter $ct_{k_v} = 0$. If an appropriate entry exists in $\mathcal{AR}$, $\mathcal{S}$ returns $str \oplus < a_1, a_2, a_3 >$ where a_1 is a random path XORed with $\mathsf{F}(sk_4, r)$, both a_2 and a_3 are $\mathsf{HE}(pk, 0)$. If no appropriate entry exists, $\mathcal{S}$ returns a random value. Then, Theorem 1 follows from the pseudo-randomness of $\mathcal{P}$ and $\mathcal{F}$ and the CPA-security of Ω.

VII. Performance Evaluation

In this section, we evaluate Gaia across four dimensions: path dilation rate, path similarity, computational cost, communication overhead. Furthermore, we compare its against SOTA Hermes.

A. Experimental Settings

Datasets and Parameters. We utilize a real-world city map of Los Angeles extracted from OpenStreetMap, comprising 2,200 intersections, 986 roads, and 6,448 road segments. The key experimental parameters are summarized as follows: the number of road network nodes N ranges from 400 to 2,200; the number of stops k ranges from 0 to 10; the Blocked Edge Rate (BER) ranges from 0% to 20%; and the number of Partial Order (PO) constraints ranges from 0 to 5. To mitigate the impact of randomness in selecting blocked edges and partial orders, we conduct each experiment five times and report the average results.

Baselines. We adopt Hermes as the primary baseline, for its superiority in sucure SCN. Since Hermes does not support blocked edges or partial order constraints, we compare it against Gaia in an ideal setting (BER=0%, PO=0). Additionally, we evaluate Gaia variants under varying BER = [5%, 15%] and PO = [1, 3, 5] to assess the algorithm's robustness in complex road conditions.

Metrics. We evaluate: (1) **Path Dilation Rate (PDR):** the ratio of the generated path length to the theoretical shortest distance, measuring distance overhead; (2) **Path Similarity (PS):** a hybrid metric combining geometric grid alignment and length ratio to evaluate trajectory quality; (3) **Computational Cost:** time consumption for offline preprocessing, token generation, and online navigation; (4) **Communication Overhead:** data volume exchanged during graph upload, query submission, and result retrieval.

Setup. The NSP is instantiated on a computer equipped with an Intel Core i7-1165G7 (2.80GHz) processor and 32GB RAM. To support high-precision geographic coordinate calculations and improve engineering efficiency, we implemented the Gaia prototype using the CKKS homomorphic encryption scheme, replacing the BGN scheme used in Hermes. CKKS natively supports floating-point arithmetic and batch processing, making it more suitable for complex road network calculations. Key parameters are set as follows: the hash

function uses SHA-256 (32 bytes), and the random oracle is instantiated with HMAC-SHA256. The CKKS ciphertext size is set to approximately 2KB to balance security and transmission efficiency, and the Garbled Circuit (GC) interaction overhead is approximately 4KB per round.

B. Path Dilation Rate

We define $PDR = |dist|/|sd|$. In the preprocessing, PDR is defined as the ratio of the path length generated by Gaia in a blocked graph to the theoretical shortest path length in an unblocked graph. Since PO are dynamic query inputs, they do not impact the static graph preprocessing. Thus, Fig. 13 evaluates preprocessing PDR against BER, k and N only.

As shown in Fig. 13, Gaia not only possesses theoretical optimality in ideal environments but also demonstrates strong geometric robustness when network topology degrades heavily. Specifically, in a blockage-free scenario (BER=0%), PDR strictly converges to 1.0, confirming that the algorithm itself introduces no extra path overhead. Facing severe road network fragmentation (BER rising to 20%), Gaia effectively controls the PDR at a low level (generally < 1.5), avoiding blind global detours. This efficiency stems from the enhanced blockage handling logic during preprocessing, which patches broken links by pre-computing local DSt, thereby exchanging minimal local cost for global connectivity. Furthermore, with increases in k and N, the PDR does not show divergent trends, and the curves for $N = 1000$ and $N = 2000$ exhibit high consistency. This superior scalability is attributed to the core MSR mechanism, which resolves complex spatiotemporal constraint conflicts via local fine-tuning.

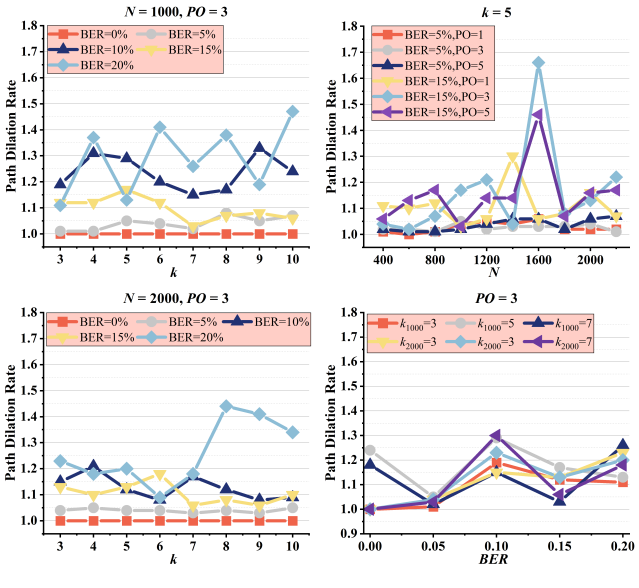

Fig. 13: Path Dilation Rate in preprocessing with varying BER.

In navigation, we define PDR as the ratio of the path length generated by Gaia to the length of the shortest path obtained from a full permutation of stops. As shown in Fig. 14, although PDR trends upward with increasing k due to the combinatorial explosion of the search space, the overall ratio is largely controlled below 1.5. Most notably, as PO increases, PDR actually decreases, approaching 1.0 at PO=5. This phenomenon profoundly validates the effectiveness of the MSR mechanism: when constraints are few (PO=0), the theoretical optimal solution often utilizes geometric advantages to significantly shorten the path, leading to a relative deviation in Gaia; however, as constraints increase, the solution space is drastically compressed, allowing Gaia to precisely lock onto the unique or few feasible solutions satisfying the temporal logic, thus achieving "geometric alignment" with the theoretical optimal solution.

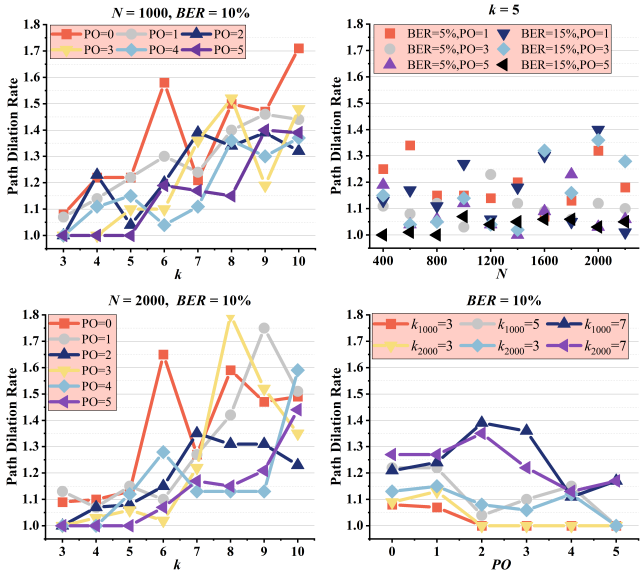

Fig. 14: Path Dilation Rate in the navigation with varying PO.

C. Path Similarity

The original node-hit ratio metric used in Hermes is no longer suitable for Gaia, because handling blocked edges and partial order constraints inevitably leads to the addition of nodes or sequence reordering. Therefore, we propose a new Hybrid Path Similarity (HPS) metric, defined as $HPS = 0.7 \times \text{GridSim} + 0.3 \times \text{LenSim}$. **Grid Similarity (GridSim):** We partition the map into uniform 50×50 meter grids and discretize the path into a grid frequency vector $V = [v_1, \ldots, [v_n]$. We calculate the spatial overlap between the actual path and the optimal path using the cosine similarity formula: $(V_{\text{act}} \cdot V_{\text{opt}})/(\|V_{\text{act}}\| \times \|V_{\text{opt}}\|)$. **Length Similarity (LenSim):** This is the ratio of the optimal distance to the actual distance. We assign a higher weight (0.7) to spatial geometric features to prioritize evaluating the path's spatial orientation and trajectory fidelity, followed by length efficiency (0.3).

The HPS in the preprocessing is shown in Fig. 15. Under the ideal baseline of $BER = 0\%$, HPS stabilizes at 100%. As the blockage rate rises to 20%, although the physical path length inevitably increases, the overall HPS maintains above 80%. This implies that the computed trajectory strictly adheres to

the optimal topology for necessary obstacle avoidance, without incurring directional deviations. Furthermore, HPS does not significantly drop with increased PO, consistent with the reasons described in the PDR analysis.

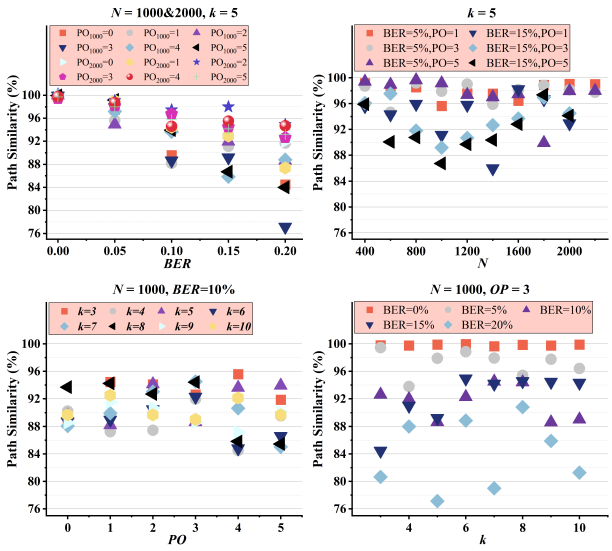

Fig. 15: Path Similarity in the preprocessing under varying BER and PO.

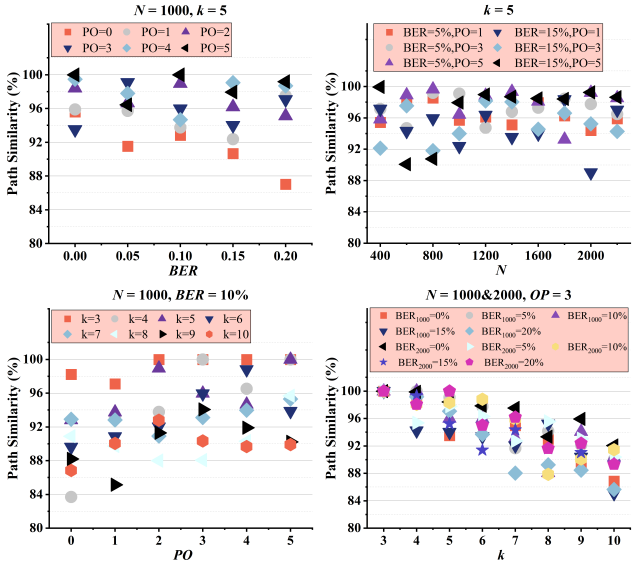

Fig. 16: Path Similarity in the navigation with varying BER and PO.

The HPS in navigation is shown in Fig. 16. Across various experimental conditions, HPS is generally guaranteed to be above 80% and mostly clusters above 90%. Although HPS decays linearly with the increase of stops k, the results remain controllable even in large-scale complex tasks with $k = 10$ and $N = 2000$. This confirms that Gaia effectively suppresses error divergence when handling long-sequence navigation tasks.

D. Computational Efficiency

We evaluate the algorithm's efficiency, including offline preprocessing overhead and online navigation time, comparing them with Hermes.

Preprocessing Efficiency. Fig. 17 illustrates the computational cost of blocked edge preprocessing. The processing time increases linearly with N. Even under extreme conditions of $N = 2200$ and $BER = 15\%$, the processing time remains approximately 550 ms, demonstrating ms-level response speed. The figure also displays the ratio of Blocked Edge Time to total initialization time. Notably, as N expands, this ratio drops sharply: from 8%-12% in small-scale networks to rapidly converging and stabilizing below 2%. This convergence highlights the algorithm's extremely low marginal cost and significant superiority in large-scale scenarios.

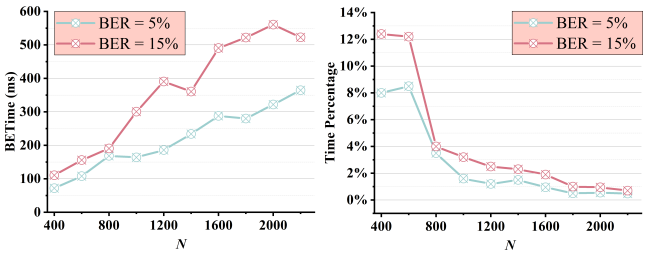

Fig. 17: Preprocessing efficiency and BeTime ratio.

Online Navigation Efficiency. When $N = 800$, the navigation time performance under different blockage rates and partial order constraints is shown in Fig. 18 and Fig. 19. Fig. 18 demonstrates a stable linear growth as k increases, confirming that Gaia successfully avoids the exponential cost of full stops permutation. This efficiency stems from our hierarchical planning strategy, which narrows the search space to local areas and determines the visitation order through geometric heuristics rather than exhaustive enumeration. Although high blockage rates inevitably increase time, performance fluctuations are minimal under low-to-medium blockage rates ($0\%-15\%$). In Fig. 19, under identical k and BER conditions, increasing the number of constraints does not significantly increase computation time. This is mainly attributed to two points: first, the MSR strategy finds safe alternative nodes in constant time, avoiding complex backtracking; second, the DaC strategy limits constraint impact to a local scope, avoiding global re-planning.

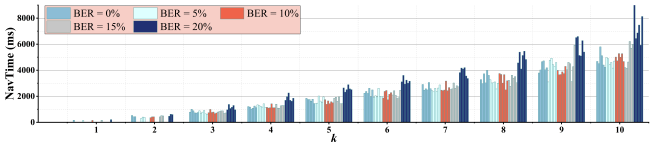

Fig. 18: Navigation time under varying BER and k ($N = 800$).

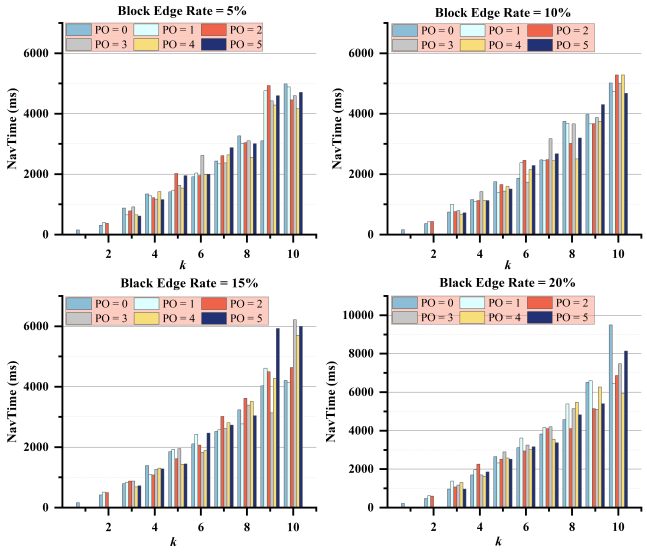

Fig. 19: Impact of Partial Order on navigation time ($N = 800$).

E. Communication and Computation Overhead

We deconstruct the total communication overhead of Gaia (C_{total}) into offline preprocessing (C_{off}) and online query (C_{on}) components. Let $|C_{\text{tx}}|$ be the ciphertext size and S_{gc} be the interaction size of a single GC round.

Offline Phase. Uploading encrypted data is a one-time cost: $C_{\text{graph}} = |PK| + 2 \cdot n \cdot |C_{\text{tx}}|$. **Online Phase.** The overhead includes token upload, result download, and interaction: $C_{\text{token}} = (k+2)(2|C_{\text{tx}}| + \delta)$, $C_{\text{res}} = (k+1)(|C_{\text{tx}}| + \gamma)$, and $C_{\text{inter}} = (PO + \alpha)S_{\text{gc}}$. Total cost scales linearly: $C_{\text{on}} = O(k|C_{\text{tx}}|) + O(PO \cdot S_{\text{gc}})$, ensuring scalability. where δ and γ are constant sizes for auxiliary identification information, and α is the base number of interaction rounds. It is evident that the online communication overhead of Gaia grows linearly with query complexity k and PO, demonstrating good scalability.

Table III shows the actual communication overhead for different numbers of stops k at a node scale of $N = 1000$.

TABLE III: Online Communication Overhead of Gaia (Unit: KB)

k	C_{token}	C_{res}	C_{inter}	C_{on}
1	12.18	6.12	8.00	26.3
2	16.25	8.25	8.00	32.5
3	20.31	10.37	8.00	38.7
4	24.37	12.50	8.00	44.9
5	28.43	14.62	8.00	51.0
6	32.50	16.75	8.00	57.2
7	36.56	18.87	8.00	63.4
8	40.62	21.00	8.00	69.6
9	44.68	23.12	8.00	75.8
10	48.75	25.25	8.00	82.0

As shown in Fig. 20, both communication and computation overheads of Gaia increase linearly with k, exhibiting excellent scalability. Even under a complex query with $k = 10$, the total communication volume is controlled at 82 KB. The

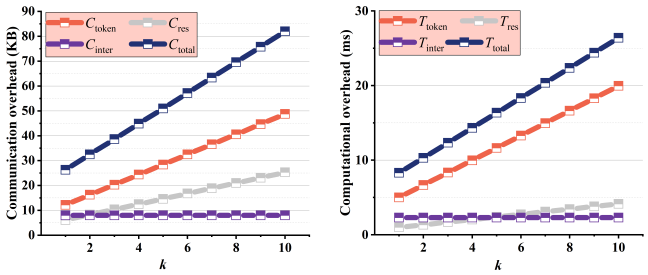

Fig. 20: Communication and computation overhead with varying k.

computational latency on the client side primarily stems from token generation. The data indicates that generating a query token for 10 stops takes less than 20 ms.

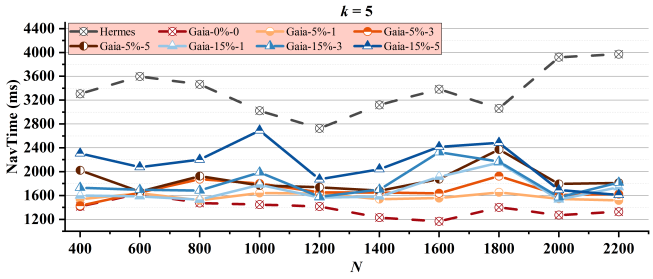

Fig. 21: Comparison of navigation efficiency between Gaia and Hermes across different N.

F. Comparison.

Finally, Fig. 21 compares the efficiency of Gaia and Hermes ($k = 5$) across varying network sizes. First, Gaia consistently outperforms Hermes. Across all test scales ($N = 400 \sim 2200$), Hermes' latency remains significantly higher (peaking near 4000 ms), while all Gaia variants lie well below it. Specifically, in the ideal case (Gaia-0%-0), Gaia maintains a time of approximately $1200 \sim 1400$ ms, achieving a substantial reduction compared to Hermes.Second, although latency increases with BER and PO, Gaia exhibits strong robustness. Even under the most stringent conditions (i.e., **Gaia-15%-5**), its time remains lower than the Hermes baseline. This indicates that Gaia successfully balances functional enhancement and speed without sacrificing system responsiveness.

VIII. Conclusions

In this work, we proposed Gaia, a secure semi-constrained navigation framework that addresses the limitations of existing schemes regarding partial visit orders and geographic obstacles. By leveraging a refined Divide-and-Conquer strategy and Homomorphic Encryption, Gaia efficiently handles Multiple Spatial Restrictions while protecting location privacy. Formal analysis proves its security under the semi-honest model. Extensive experiments on real-world datasets demonstrate that Gaia significantly outperforms the SOTA baseline, Hermes.Furthermore, the evaluation results confirm that Gaia

maintains precise **geometric alignment** with optimal trajectories, exhibiting robustness and scalability even in complex and obstacle-dense environments.

REFERENCES

[1] S. Lai, X. Yuan, S.-F. Sun, J. K. Liu, Y. Liu, and D. Liu, "GraphSE^2: An encrypted graph database for privacy-preserving social search," *Proc. 14th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, July 2019: 41-54, Auckland, New Zealand.

[2] Y. Peng, Z. Ma, W. Zhang, X. Lin, Y. Zhang, and X. Chen, "Efficiently answering quality constrained shortest distance queries in large graphs," *Proc. 39th IEEE International Conference on Data Engineering (ICDE)*, 2023: 856-868, Anaheim, USA.

[3] T. Yu, J. Liu, Y. Yang, Y. Li, H. Fei, and P. Li, "EGM: Enhanced Graph-based Model for Large-scale Video Advertisement Search," *Proc. 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2022: 4443 - 4451, Washington, USA.

[4] P. Zhao, J. Yu, H. Zhang, Z. Qin and C. Wang, "How to securely outsource finding the min-cut of undirected edge-weighted graphs," *IEEE Transactions on Information Forensics and Security (TIFS)*, 2020, 15: 315-328.

[5] K.-T. Foerster, J. Hirvonen, Y.-A. Pignolet, S. Schmid and G. Tredan, "On the Price of Locality in Static Fast Rerouting," *Proc. 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022: 215 - 226, MD, USA.

[6] Z. Liu, S. Shekhar and C. Peng, "Breaking Geographic Routing Among Connected Vehicles," *Proc. 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2023: 42 - 54, Porto, Portugal.

[7] TechCrunch, "The biggest data breaches in 2024: 1 billion stolen records and rising," https://techcrunch.com/2024/10/14/2024-in-data-breaches-1-billion-stolen-records-and-rising, 2024.

[8] HealthCare IT News, "Oracle Health customers notified of data compromise, reports say," https://www.healthcareitnews.com/news/oracle-health-customers-notified-data-compromise-reports-say, 2025.

[9] X. Meng, S. Kamara, K. Nissim, and G. Kollios, "GRECS: Graph encryption for approximate shortest distance queries," *Proc. 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, October, 2015: 504-517, Denver, USA.

[10] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell, "Privacy-preserving shortest path computation," *Proc. 23rd Annual Network and Distributed System Security Symposium (NDSS)*, February, 2016, San Diego, USA.

[11] S. Wang, Y. Zheng, X. Jia, and X. Yi, "PeGraph: A system for privacy-preserving and efficient search over encrypted social graphs," *IEEE Transactions on Information Forensics and Security (TIFS)*, 2022, 17: 3179-3194.

[12] J. G. Chamani, I. Demertzis D. Papadopoulos, C. Papamanthou, and R. Jalili, "GraphOS: Towards Oblivious Graph Processing," *Proc. 49th International Conference on Very Large Data Bases (VLDB)*, 2023, 16(13): 4324 - 4338, 2023.

[13] New York Times, "Twelve Million Phones, One Dataset, Zero Privacy," https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html, 2019.

[14] Q. Wang, K. Ren, M. Du, Q. Li, and A. Mohaisen, "Secgdb: Graph encryption for exact shortest distance queries with efficient updates," *Proc. 21st International Conference on Financial Cryptography and Data Security (FC)*, April 2017: 3-7, Sliema, Malta.

[15] E. Ghosh, S. Kamara, and R. Tamassia, "Efficient graph encryption scheme for shortest path queries," *Proc. 16th ACM Asia Conference on Computer and Communications Security (ASIACCS)*, June 2021: 516-525, Hong Kong, China.

[16] M. Du, S. Wu, Q. Wang, D. Chen, P. Jiang, and A. Mohaisen, "Graphshield: Dynamic large graphs for secure queries with forward privacy," *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 2022, 34 (7): 3295-3308.

[17] F. Wang, Z. Chen, L. Pan, L. Y. Zhang, and J. Zhou, "CryptGraph: An efficient privacy-enhancing solution for accurate shortest path retrieval in cloud environments," 'emphProc. 19th ACM Asia Conference on Computer and Communications Security (ASIACCS), 2024, 1660-1674, Singapore, Singapore.

[18] F. Falzon, E. Ghosh, K. G. Paterson, and R. Tamassia, "PathGES: An efficient and secure graph encryption scheme for shortest path queries," *Proc. 31st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, October 2024: 4047-4061, Salt Lake City, USA.

[19] C. Liu, L. Zhu, X. He and J. Chen, "Enabling privacy-preserving shortest distance queries on encrypted graph data," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2021, 18 (1): 192-204.

[20] M. Li, Y. Chen, J. Gao, J. Wu, Z. Zhang, J. He, L. Zhu, M. Conti, and X. Lin, "Accurate, secure, and efficient semi-constrained navigation over encrypted city maps," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2024, Online. DOI: 10.1109/TDSC.2024.3521396.

[21] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," *Proc. 23rd International Conference on the Theory and Applications of Cryptology and Information Security (ASIACRYPT)*, December, 2017: 409-437, Hong Kong, China.

[22] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," *Proc. 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, December 2010: 577-594, Singapore.

[23] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-hop labels," *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2002: 937-946, San Francisco, USA.

[24] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *Proc. 13th ACM conference on Computer and Communications Security (CCS)*, October 2006: 79-88, Alexandria, USA.

[25] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," *Proc. 16th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2010: 577-594, Singapore.

[26] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," *Proc. 23rd International Conference on the Theory and Applications of Cryptology and Information Security, (ASIACRYPT)*, December 2017: 409-437, Hong Kong, China.

[27] J. Katz and Y. Lindell, "Introduction To Modern Cryptography (Third edition)," *Chapman & Hall/CRC Cryptography and Network Security (51 books)*, 2021.

[28] A. Yao, "Protocols for secure computations (extended abstract)," *Proc. 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, November 1982: 160-164, Chicago, USA.

[29] A. Yao, "How to generate and exchange secrets (extended abstract)," *Proc. 27th Annual Symposium on Foundations of Computer Science (FOCS)*, October 1986: 162-167, Toronto, Canada.

[30] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," *Proc. 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January, 2001: 448-457, Washington, USA.

[31] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," *Proc. 2nd Theory of Cryptography, Second Theory of Cryptography Conference (TCC)*, February 10-12, 2005: 325-341, Cambridge, USA.

[32] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, 1962, 5 (6): 344-348.