ASTRO: Reducing COVID-19 Exposure through Contact Prediction and Avoidance

CHRYSOVALANTIS ANASTASIOU, University of Southern California
CONSTANTINOS COSTA and PANOS K. CHRYSANTHIS, University of Pittsburgh
CYRUS SHAHABI, University of Southern California
DEMETRIOS ZEINALIPOUR-YAZTI, University of Cyprus, Cyprus

The fight against the COVID-19 pandemic has highlighted the importance and benefits of recommending paths that reduce the exposure to and the spread of the SARS-CoV-2 coronavirus by avoiding crowded indoor or outdoor areas. Existing path discovery techniques are inadequate for coping with such dynamic and heterogeneous (indoor and outdoor) environments—they typically find an optimal path assuming a homogeneous and/or static graph, and hence they cannot be used to support contact avoidance. In this article, we pose the need for *Mobile Contact Avoidance Navigation* and propose *ASTRO* (Accessible Spatio-Temporal Route Optimization), a novel graph-based path discovering algorithm that can reduce the risk of COVID-19 exposure by taking into consideration the congestion in indoor spaces. *ASTRO* operates in an A^* manner to find the most promising path for safe movement within and across multiple buildings without constructing the full graph. For its path finding, *ASTRO* requires predicting congestion in corridors and hallways. Consequently, we propose a new grid-based partitioning scheme combined with a hash-based two-level structure to store congestion models, called *CM-Structure*, which enables on-the-fly forecasting of congestion in corridors and hallways. We demonstrate the effectiveness of *ASTRO* and the accuracy of *CM-Structure*'s congestion models empirically with realistic datasets, showing up to one order of magnitude reduction in COVID-19 exposure.

CCS Concepts: • **Information systems** → **Mobile information processing systems**; *Data mining*; **Spatial-temporal systems**; • **Computing methodologies** → *Machine learning;*

Additional Key Words and Phrases: Indoor, outdoor, path recommendation, graph processing, disability, congestion forecasting

11

Chrysovalantis Anastasiou and Constantinos Costa contributed equally to the article.

This work was partially funded by NIH award U01HL137159, by the Pittsburgh Foundation, by NSF grants CNS-2027794 and CNS-2125530, and an unrestricted cash gift from Microsoft Research. The last author's research has been supported by EUs H2020-EU.3.4 LASH FIRE project, under grant agreement No 814975; EUs H2020 MSCA RISE RESPECT project, under grant agreement No 101007673; and Cyprus Research Promotion Foundation RESTART programme, under project EnterCY INTEGRATED/0609/0020.

Authors' addresses: C. Anastasiou, University of Southern California, Los Angeles, California, USA, 90089; email: canastas@usc.edu; C. Costa, University of Pittsburgh, Pittsburgh, Pennsylvania, USA, 15260; email: costa.c@cs.pitt.edu; P. K. Chrysanthis, University of Pittsburgh, Pittsburgh, Pennsylvania, USA, 15260; email: panos@cs.pitt.edu; C. Shahabi, University of Southern California, Los Angeles, California, USA, 90089; email: shahabi@usc.edu; D. Zeinalipour-Yazti, University of Cyprus, Nicosia, Cyprus; email: dzeina@cs.ucy.ac.cy.

2374-0353/2021/12-ART11
https://doi.org/10.1145/3490492

ACM Reference format:
Chrysovalantis Anastasiou, Constantinos Costa, Panos K. Chrysanthis, Cyrus Shahabi, and Demetrios Zeinalipour-Yazti. 2021. ASTRO: Reducing COVID-19 Exposure through Contact Prediction and Avoidance. *ACM Trans. Spatial Algorithms Syst.* 8, 2, Article 11 (December 2021), 31 pages.
https://doi.org/10.1145/3490492

1 INTRODUCTION

The COVID-19 pandemic dramatically impacted the mobility behavior of people, from indoor and outdoor pedestrian movement to public transportation and ridesharing [22]. COVID-19 is highly *contagious* and a high percentage of infected people do not initially exhibit symptoms, making **Contact Tracing (CT)** a crucial task [4]. The biggest problem with human-based CT is that it is relatively slow (i.e., high latency) and cannot cope with the speed at which the virus spreads under loose distancing measures [18]. Thus, governments showed interest in building computerized systems and applications that can automate the contact tracing process and address the challenges of the high latency tracing procedure and massive human effort. However, the so-called ***Mobile Contact Tracing Applications (MCTAs)*** [18, 52] are not producing the expected results because of low participation rates and a lack of user trust due to privacy concerns [37, 51]. Furthermore, MCTAs only focus on detecting the spread of the SARS-CoV-2 coronavirus, but not on reducing the risk of COVID-19 exposure of individuals.

We clearly need preventive measures to reduce the risk of COVID-19 exposure to people. The exposure risk can be reduced by observing social distancing, avoiding public transportation and ridesharing, and as long as weather conditions permit, by walking through outdoor paths. However, during winter, with subfreezing temperatures (as seen in Figure 1 - right), and during summer, with extreme heat and poor air quality (as seen in Figure 1 - left), minimizing the outdoor exposure is a necessity and indoor paths are desirable. Thus, indoor paths are desirable, if not unavoidable, and moving indoors through congested pathways dramatically increases the risk of exposure to airborne viruses[1] (as seen in Figure 1 - center). According to the **US Centers for Disease Control and Prevention (CDC)**, human coronaviruses and influenza, like any other crowd disease, most commonly spread from an infected person to others through the air by coughs, sneezes, and close personal contact, such as touching or shaking hands. That is, walking along congested pathways remains one of the most common ways for a person to contract the SARS-CoV-2 coronavirus. This clearly highlights the importance of *contact avoidance*, besides contact tracing, in the fight against the COVID-19 pandemic, and the need for ***Mobile Contact Avoidance Navigation (MCAN)*** solutions that avoid crowded indoor hallways and indoor spaces. Posing the need for MCAN constitutes the broader impact of this article.

MCAN solutions must satisfy the following four criteria to be effective and efficient: (i) be inclusive (i.e., consider with mobility barriers); (ii) be context-aware (i.e., adapt its recommendation at any given time based on the current situation); (iii) be location-aware (i.e., deal with the congestion in specific locations); and (iv) be practical (i.e., provide a holistic/end-to-end path.)

As these four criteria are not conflicting requirements, multi-objective navigation solutions are not applicable. Furthermore, for most current navigation solutions, these criteria focus on indoor or outdoor routing. Only a handful of the state-of-the-art solutions consider indoor-outdoor seamless transition techniques [34], but not a *unified model* that can support *end-to-end* path search [23]. Moreover, existing indoor solutions cannot support contact avoidance and reduce the risk of

[1]Centers for Disease Control and Prevention: https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/how-covid-spreads.html.

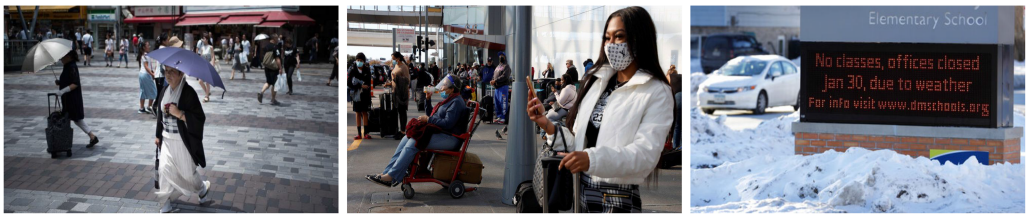

Fig. 1. (left) The public is being advised to take every precaution to avoid the extreme heat in Japan (BBC 2018); (center) US airports experienced their highest passenger numbers during Thanksgiving holiday since March 2020 (BBC 2020); (right) An elementary school closed due to cold weather in Des Moines, Iowa (CNN 2019).

COVID-19 exposure by being oblivious to the congestion and movement directionality in corridors and hallways [41]. These solutions typically find an optimal path by assuming a homogeneous and/or static graph and cannot operate over *time-varying* dynamic graphs [20], which are inherent in the presence of congestion that changes over time.

Dynamically constructing a graph for path discovery with multiple constraints is very expensive, especially when the graph is complete (fully connected), which should be expected in MCAN solutions. Considering a real scenario where all outdoor vertices V_o (i.e., buildings) are connected using all indoor vertices V_i (i.e., doors) the complexity for the construction is $O(|V_i|^2)$, while the maximum number of edges are $|E| = \frac{|V_i|*(|V_i|-1)}{2} = O(|V_i|^2)$. The complexity for finding the path with a well-known algorithm such as Dijkstra is $O(|V_i|^2 + |V_i| * log|V_i|)$. Whereas, the complexity of the state-of-the-art temporal path discovery algorithm in a dynamic graph described in Reference [54] is $O(|V_i| + |V_i|^2 * log|V_i|)$. To the best of our knowledge, none of the existing path-finding algorithms can efficiently discover optimal paths with constraints in an indoor-outdoor complete graph to meet MCAN requirements. Therefore, in this work, we address the *problem of fast and incremental temporal path discovery for an indoor-outdoor graph.*

In this article, we propose ***ASTRO* (Accessible Spatio-Temporal Route Optimization)** for fast and incremental temporal path discovery for an indoor-outdoor graph. *ASTRO* is a novel graph-based path-finding algorithm that takes into consideration indoor congested spaces and can reduce the risk of COVID-19 exposure. *ASTRO* can operate over time-varying dynamic graphs without constructing the complete graph. It integrates outdoor nodes (i.e., buildings) with indoor nodes (e.g., doors, stairs, escalators, elevators) to efficiently provide a personalized contact-avoidance path satisfying a user-specified set of constraints (i.e., accessibility requirements, congestion tolerance, arrival time, and outdoor exposure). *ASTRO* uses *time* to innovatively unify indoor congestion and travel distance in an A^* search with a complexity of $O(|V_i|^2)$. However, our experiments over real datasets verify that the worst-case scenario will rarely happen due to the greedy A^* search reduction approach. The congestion of each indoor path segment is predicted based on the estimated arrival time at a segment and converted into travel time slowdown (i.e., delay).

ASTRO's second innovation is a new grid-based partitioning scheme, named *Building Grid Layout*, combined with a multilevel access method, called ***Congestion Model Structure* (*CM-Structure*)**, which supports efficient retrieval of the congestion prediction models during the path finding. *CM-Structure* is also used to store the trained forecasting models of each corridor and hallway segment. We use machine learning to build a congestion model for each indoor segment (i.e., corridor or hallway).

We evaluate our algorithms experimentally using two realistic datasets consisting of buildings at the University of Pittsburgh and University of Cyprus. Our experiments show that a ***Fully Connected Recurrent Neural Network* (FC-RNN)** is most suitable to be used for congestion

prediction, since it offers very low inference time while requiring less memory than other commonly used models. Additionally, FC-RNN is proven to support continuous (lifelong) learning [14] so models can be re-trained in an online fashion to keep up with the ever-changing dynamics of the congestion. Similarly, *CM-Structure* with hash access method provides the most efficient storage and retrieval of the congestion models relevant to an indoor path. Our experiments also show that *ASTRO* using *CM-Structure* and FC-RNN can reduce congestion by up to one order of magnitude while incurring low response time.

Towards developing the first *MCAN* solution, the contributions of this article are summarized as follows:

- We propose a novel efficient routing algorithm, dubbed *ASTRO*, for contact avoidance in which all combinations of all building's entrances and exits are instantiated as separate nodes in an incrementally created search graph.
- We propose the use of prediction models that support continuous learning to forecast congestion in indoor spaces.
- We propose a novel partitioning scheme and algorithm, called *Building Grid Layout*, that works in conjunction with *CM-Structure*, a new access method to organize the trained congestion models, enabling on-the-fly congestion forecasting in hallways and corridors.
- We propose a method to generate realistic indoor congestion datasets for experimental evaluation when real datasets are not available.
- We measure the efficiency of *ASTRO* and the congestion models using two realistic datasets, showing that *ASTRO* produces an optimal path in terms of both satisfying the user-provided constraints and minimizing the travel time, while reducing COVID-19 exposure by up to one order of magnitude.

The remainder of the article is structured as follows: Section 2 formulates the problem and introduces our novel *ASTRO* algorithm. Section 3 describes the procedures we employ to accurately forecast the congestion in corridors. Section 4 outlines the steps we employ to generate realistic indoor congestion datasets. Section 5 presents an experimental evaluation of our proposed *ASTRO* algorithm and the congestion predictive models. Section 6 discusses related work and Section 7 concludes the article and discusses future work.

2 CONTEXT-AWARE PATH RECOMMENDATION FOR CONTACT AVOIDANCE

In this section, we formulate the ***Mobile Contact Avoidance Navigation*** **(MCAN)** problem and present our solution. First, we formalize our system model and underlying assumptions of our solution (Section 2.1) and then introduce our *ASTRO* algorithm (Section 2.2) and its implementation details (Section 2.3).

2.1 Problem Formulation

The objective of MCAN is to find a safe path between two locations for a given individual. Such a safe personalized path considers the preferences or constraints of an individual, which include any accessibility restrictions, departure and arrival time requirements, time spent outdoors at any given period, and congestion tolerance. Congestion tolerance is the degree/level of congestion that an individual can navigate through a crowded space with significantly reduced COVID-19 exposure risk. This risk depends on the vaccination and the masking of an individual. According to CDC, staying over 15 minutes in an indoor, congested space dramatically increases the COVID-19 exposure risk, depending on the quality of the mask. The median type of mask used provides a squared protection factor of the spreader and infectee PF2 =2 (reducing infection virus inhalation by only 51%). In contrast, the use of available higher-quality masks (KN95) produces a PF2 =400;

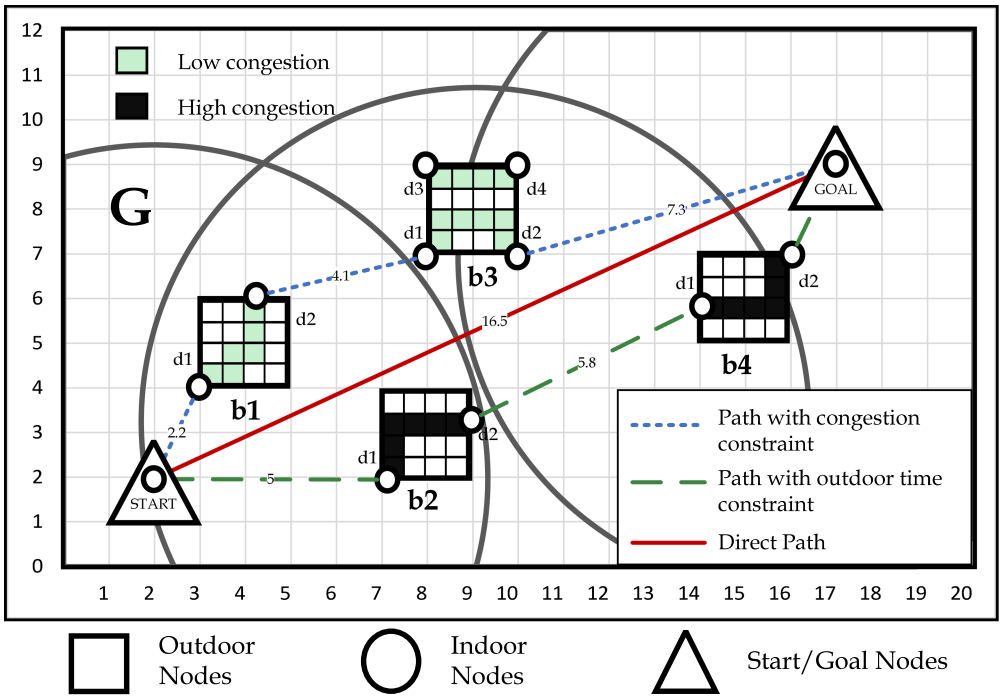

Fig. 2. *ASTRO* is an algorithm that can recommend a path with the limited outdoor exposure and congestion, while keeping the travel time low for each request along with the corresponding source and final destination. The red solid line represents a path without any constraints, the green dashed line represents a path with 7.5 minutes outdoor exposure limit, and the dotted blue path avoids the buildings $b2$ and $b4$, which are highly congested, marked in black color.

a reduction of 99.75% or 200× reduction in airborne infection when compared to the median cloth mask. Consequently, an individual wearing a KN95 mask can specify high congestion tolerance.

We are assuming the typical graph representation of urban topology $G(B \cup D, E_o \cup E_i)$ consisting of outdoor vertices B representing buildings enhanced with the indoor vertices D representing entrances or exits (e.g., doors), as shown in Figure 2. Within each building edges e_i ($e_i \in E_i$) are corridors connecting entrances and exits. Each corridor is segmented into fixed area cells with predefined maximum density (shown in color/shaded in Figure 2). The actual density of a cell on a given date and time quantifies the cell's congestion at that timestamp.

The constraints that defined the context of MCAN and our solution are as follows. The notation is summarized in Table 1.

Definition 2.1. ***Accessibility (A)*** is a binary constraint that represents the requirement for handicap-accessible doors and corridors. For example, sufficiently wide doors (both into and in the building), with easily accessible automatic door openers and ramps.

Definition 2.2. ***Outdoor Exposure Limit (OE)*** is the duration in seconds of a path or segment of a path that is exposed to outdoors conditions, i.e., $OE \in [0, \infty)$. Specifically, if $OE = 0$, then only indoor paths will be considered. In contrast, if $OE = \infty$, then the constraint is ignored.

Definition 2.3. ***Congestion Tolerance Limit (CT)*** is the percentage of the density (i.e., number of people in a unit of indoor space), which is acceptable to go through a hallway or a corridor. For

Table 1. Summary of Notations

Notation	Description
B	set of all outdoor vertices (e.g., buildings) $j = 1, \ldots, n$
b_i	an outdoor vertex $b_i \in B$
D	set of all indoor vertices (e.g., door) $l = 1, \ldots, m$
d_i	indoor vertex $d_i \in D$ for an outdoor vertex b
t	travel time
o/OE	outdoor exposure/outdoor exposure limit
c/CT	congestion/congestion tolerance limit
A	accessibility
θ	set of constraints (A, OE, CT)
M	indexed set of the models

example, a congestion tolerance limit of 50% ($CT = 0.5$) in 9 m^2 (3×3 m) indoor space will be 4.5, assuming that an individual occupies 1 m^2.

These three constraints are sufficient to express a wide range of preferences for pedestrians. For example, a pedestrian can: (i) set lower outdoor exposure tolerance when there is a heatwave or polar vortex; (ii) set higher congestion tolerance if vaccinated or wears KN95 mask; and (iii) enable accessibility when they prefer to avoid stairs, revolving doors, and so on. Another example beyond contact-avoidance of the generality of *ASTRO* with the three constraints is the use of congestion tolerance to address social anxiety and overstimulation conditions.

Research Goal: *Given a set of buildings (i.e., outdoor vertices) and their floor plans (e.g., corridors, doors, and rooms) along with an origin and a destination point,* ASTRO *aims to produce a path with the minimal total travel time from the origin to the destination that satisfies the* ***accessibility constraint*** *(A), the* ***outdoor exposure limit*** *(OE), and the* ***congestion tolerance limit*** *(CT).*

The effectiveness of our proposed technique in achieving the above goal is measured as follows:

Definition 2.4. ***Total Travel Time (T)*** is the total travel time of the path between the source and the destination point. This is the sum of the total outdoor exposure time (i.e., the duration of the path that is exposed to outdoors conditions) and the total indoor time (i.e., the duration of the path that is indoors). The indoor time includes the slowdown (i.e., delays) in going through a congested hallway or corridor.

Definition 2.5. ***Congestion Reduction (C)*** is the indoor congestion of the path between the source and the destination point compared to a shortest path oblivious to congestion.

ASTRO is a single-objective (i.e., minimizing T) and multi-constrained (i.e., satisfying the set of constraints $\{A, OE, CT\}$) path-finding algorithm. Particularly, *ASTRO* discovers a *feasible* path (with respect to the aforementioned constraints) that is *optimal* path in terms of total travel time.

2.2 The *ASTRO* Algorithm

As mentioned above, the *ASTRO* algorithm behaves as an A^* traversal algorithm on a time-varying graph in which the unit of cost is *time*—both the distance covered $g(n)$ (*gScore*) from start until the current node n, which includes indoor congestion delays, and the heuristic function $h(n)$ of the remaining distance to the destination are expressed in terms of time. The time is calculated based on the average walking speed 1.4 m/s [6, 24]. *ASTRO* avoids the expensive construction

of a complete graph G by pruning the edges and paths that do not satisfy the constraints $\theta = (A, OE, CT)$ of accessibility, outdoor exposure, and congestion tolerance limits, and incrementally explores the constraint-satisfying search space (graph G') to find the best path.

Specifically, the algorithm integrates the outdoor and indoor travel time contributions as a weight on the outdoor edges (i.e., outdoor paths connecting pairs of buildings). That is, as shown in Figure 2, the weight $w_{1.2,3.1}$ of the edge from b_1 to b_3 records the time o to reach from building b_1 to building b_3, plus the indoor time i between the entrance door d_1 and exit door d_2 in b_3. In other words, $w_{1.2,3.1}$ is the total time from the exit door d_2 of b_1 to exit door d_2 of b_3 plus any delay due to congestion c (i.e., $t = o + i + i * c$). *ASTRO*'s innovation is the time-varying exploration of every combination of an entrance d_i and an exit d_j of a building b as a separate node b_{d_i,d_j} in G'.

The indoor travel time i is calculated using the indoor distance provided by Anyplace [53] and the average walking speed. As indicated above, *ASTRO* calculates the delay due to congestion c as a percentage of the original indoor travel time. For example, if the original indoor time i = 10 s and there is 50% congestion, then the congestion delay is 5 s. The congestion is predicted for each segment of the indoor path using the *CM-Structure* described in Section 3.

At each iteration until reaching the destination, *ASTRO* visits all nodes b_{d_i,d_j} that satisfy the OE constraint (shown with circles in Figure 2) and have d_i and d_j satisfying the A constraint. *ASTRO* prunes out those that do not satisfy the CT constraint and selects the node with the minimum $f(n)$ ($fScore$) to expand the path. The $fScore$ of a node n is defined as follows:

$$f(n) = g(n) + h(n)$$
$$where\ g(n) = \sum_{\forall pairs\ b_k.d_j, b_l.d_i\ \in\ current\ path} w_(b_k.d_j, b_l.d_i). \quad (1)$$

ASTRO uses a priority queue, named *OPEN*, to sort the nodes based on $fScore$ and one set, named *CLOSED*, to store the visited nodes.

Theorem 2.1. *ASTRO is optimal.*

Proof. *ASTRO* as an A^* variant is optimal [21]. *ASTRO* (Algorithm 2) satisfies the two criteria of A^* optimality:

(1) *ASTRO* expands only nodes whose f-values are less (or equal) to the optimal cost path P* ($f(n)$ is less-or-equal than P*).
(2) The evaluation function of a goal node along an optimal path equals P*. □

Example: To illustrate *ASTRO* in action, consider again the set of buildings in Figure 2 where a user wants to travel from a source $start$ to the destination $goal$. Both $start$ and $goal$ are outdoor vertices with only one indoor vertex. For sake of simplicity, the outdoor vertices representing buildings correspond to only unidirectional pairs of indoor vertices (doors) that define an indoor path (e.g., only $b1_{d1,d2}$ is added into the queue, but not $b1_{d2,d1}$). Additionally, each metric unit on Figure 2 maps to one minute.

First, the $gScore$ and $fScore$ are computed for the $start$ vertex ($start(g = 0, f = 16.6)$) and then $start$ is placed into *OPEN*. Then *ASTRO* expands all neighboring nodes of $start$ calculating the weights, $gScore$, and $fScore$. We illustrate how the next steps change: (i) when a constraint of 450 s outdoor exposure limit is specified; and (ii) when an additional constraint of 0% congestion tolerance limit (i.e., avoid the congested buildings) is specified. Particularly:

(i) With 450 seconds outdoor exposure limit: The *OPEN* queue changes to $\{b2_{d1,d2}(g = 8, f = 18), b1_{d1,d2}(g = 6.2, f = 19.4))\}$ and the *CLOSED* set to $\{start(g = 0, f = 16.6)\}$, as shown in Figure 3 (Iteration 1 - left). This happens because only the $b2$ and $b1$ satisfy the 450 seconds constraint

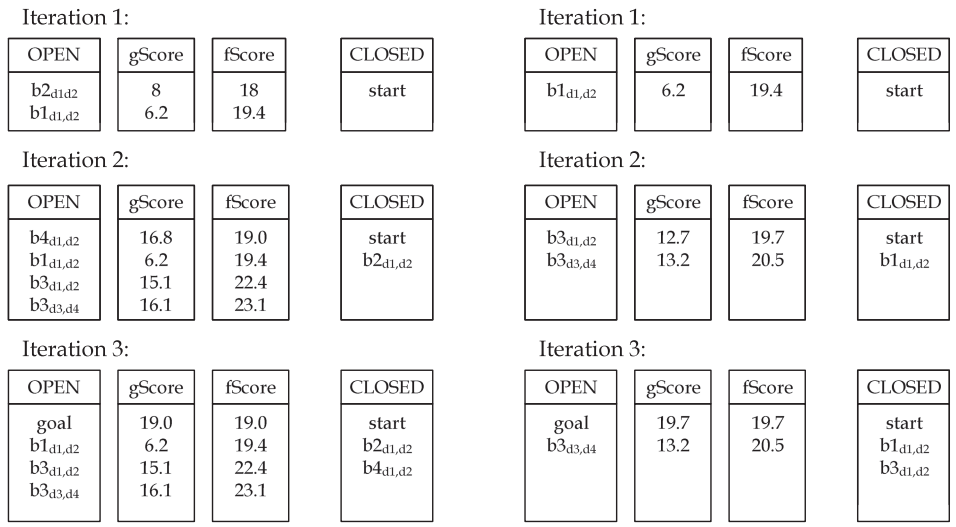

Fig. 3. (left) *ASTRO* execution example when a constraint of 450 s outdoor exposure limit is specified and (right) when an additional constraint of 0% congestion tolerance limit is specified.

starting from the *start* node. In the second iteration, the first element from the *OPEN* queue is dequeued, which is $b2_{d1,d2}(g = 8, f = 18)$ and the weight, *gScore*, and *fScore* for each of its neighbor is calculated. Specifically, the *gScore* and *fScore* are calculated for $b1$, $b3$, $b4$, and *goal* resulting in the addition only of $b4$ and $b3$ to the *OPEN* queue, as shown in Figure 3 (Iteration 2 - left). The *goal*'s outdoor time from $b2$ is more than 450 seconds, thus it is not enqueued. In the third iteration, $b4_{d1,d2}(g = 16.8, f = 19.0)$ is dequeued and the weights and scores for $b1$, $b3$, and *goal* are calculated. This results in an updated *OPEN* queue and *CLOSED* set as shown in Figure 3 (Iteration 3 - left). In the last iteration, *goal* is dequeued, the search stops (line 7), and the path is reconstructed (shown as a green dashed line in Figure 2).

(ii) With 450 seconds outdoor exposure limit and congestion tolerance limit: The priority queue *OPEN* changes to $\{b1_{d1,d2}(g = 6.2, f = 19.4)\}$ and the *CLOSED* set is $\{start(g = 0, f = 16.6)\}$, as shown in Figure 3 (Iteration 1 - right). This happens because only the $b1$ satisfies the constraints from the *start* node. In the second iteration, the first element from the *OPEN* queue is dequeued again, which is $b1_{d1,d2}(g = 6.2, f = 19.4)$ and the weight, *gScore*, and *fScore* for each of its neighbors are calculated. Specifically, the *gScore* and *fScore* are calculated for $b3$, and *goal* resulting in the addition only of $b3$ to the *OPEN* queue, as shown in Figure 3 (Iteration 2 - right). The outdoor time from $b1$ to *goal* is more than 450 seconds, and $b2$ and $b4$ are highly congested and hence $b1$, $b2$, and $b3$ are pruned out. In the third iteration, $b3_{d1,d2}(g = 12.7, f = 19.7)$ is dequeued and the weights and scores for *goal* are recalculated. This results in an updated *OPEN* queue and *CLOSED* set, as shown in Figure 3 (Iteration 3 - right). In the last iteration, *goal* is dequeued, the search stops (line 7), and the path is reconstructed (shown as a blue dotted line in Figure 2).

2.3 The *ASTRO* Implementation

This section describes the details of the implementation along with the internal data structures used in *ASTRO*. To guarantee the correctness of the path, *ASTRO* uniquely determines an outdoor vertex based on the building identification code and the indoor path, i.e., the pair of the indoor vertices. For example, b_3 with $\{d_1,d_2\}$ is different than b_3 with $\{d_1,d_3\}$. The indoor vertices d_is encode their geographical coordinates, i.e., longitude and latitude.

ALGORITHM 1: - Calculate Status Function used in Algorithm 2

Input: *in*: indoor vertex; *out*: indoor vertex; *time*: travel time;
Output: $\vec{s}$: status;

```
1:  function CalcStatus(in, out, time)
2:      if (in, out) in the same building then                ▷ If the path is indoors
3:          congestion ← PredictCongestion(CM-Structure, in, out, time)
4:          indoor ← Distance(in, out) ÷ avg_velocity
5:          outdoor ← 0
6:      else
7:          congestion ← 0
8:          indoor ← 0
9:          outdoor ← Distance(in, out) ÷ avg_velocity
10:     end if
11:     time ← outdoor + indoor + indoor ∗ congestion
12:     s⃗ ← (congestion, outdoor, indoor, time)
13:     return s
14: end function
```

The outdoor vertices b_i and b_j are connected using their selected indoor vertices $d_k \in b_i$ and $d_l \in b_j$, which are the entrances or the exits of the buildings. The *status* of a building b_j is a vector of values, i.e., $\vec{s} = \{c, i, o, t\}$, that records the congestion c (i.e., minimum, average, and maximum congestion) in b_j, the time spent indoors i in b_j, the time spent outdoors o to reach b_j, and the total travel time t ($= i + o + i * c$). Note that the status of b_j implements the concept of the edge $b_i.d_k$ to $b_j.d_l$ with its weight in search graph G' described above.

The status vector is computed by the `CalcStatus` function, which is shown in Algorithm 1. `CalcStatus` accepts as inputs a pair of indoor vertices *in* and *out* and the current total travel time since start time. If *in* and *out* are in the same building (lines 2–5), then the indoor travel time i is calculated by using the function `Distance` that retrieves the shortest indoor path *in-out* from Anyplace [29, 53] and the average walking speed [6, 24]. `PredictCongestion` is invoked to calculate the percentage of congestion c using the *CM-Structure* described in Section 3. The total time t for the pair of vertices is the sum of the indoor time plus the delay due to congestion. The delay due to the congestion is calculated as *indoor time* $*$ *congestion* [2] (line 11). That is, if *in* and *out* are in the same building, then it returns the status $\vec{s} = \{c, i, 0, t\}$ (line 12). If the vertices *in* and *out* belong to two different outdoor vertices (lines 6–10), then the path is outdoors and, therefore, the indoor travel time and the congestion are set to 0, and $t = o$. That is, the return status $\vec{s} = \{0, 0, o, t\}$.

The *ASTRO* algorithm pseudocode is shown in Algorithm 2. In step 1, the priority queue *OPEN* and the set of visited outdoor vertices *CLOSED* are initialized. In the graph construction and path finding step (Step 2 - lines 3–36), the source outdoor vertex *start* is initialized and enqueued into *OPEN*.

While the *OPEN* is not empty, the top outdoor vertex in *OPEN* is dequeued into *current*. If *current* is the same with the *goal*, then the destination was reached and the path p is reconstructed using the `ReconstructPath` function (shown in the Algorithm 3) and terminates. If *current* is not in the *CLOSED* set, then the algorithm checks all the indoor paths (e.g., pairs of indoor vertices) by iterating over the indoor vertices in b for all of *current*'s neighbors in B set (line 13).

ALGORITHM 2: - *ASTRO: Graph Integrator and Path Finder*

Input: $start$: source; $goal$: destination; B: outdoor vertices; θ: set of Constraints;
Output: Recommended path p;

▷ **Step 1: Initialization**
$OPEN \leftarrow \emptyset$ ▷ Create a priority queue
$CLOSED \leftarrow \emptyset$ ▷ Create an empty set of the visited nodes
▷ **Step 2: Graph construction and Path Finding**
$start[gScore] \leftarrow 0$
$start[fScore] \leftarrow start[gScore] + h(b[exit], goal)$
$start[camefrom] \leftarrow \emptyset$
Enqueue($OPEN, start$) ▷ The priority queue ($OPEN$) is based on $fScore$
while $OPEN$ not empty **do**
  $current \leftarrow Dequeue(OPEN)$
  if $current = goal$ **then**
    $p \leftarrow$ ReconstructPath($current$) ▷ Stop looping and return the path
  end if
  if $current \notin CLOSED$ **then**
    for all $b \in B$ **do** ▷ for each neighbor of $current$
      if $current \neq b$ and $b \notin CLOSED$ **then** ▷ Find the entrance (in) and exit (out) of b
        for all $in \in b$ **do**
          ▷ $s1$ is the status of the outdoor path
          $\vec{s1} \leftarrow CalcStatus(current[exit], in, current[gScore])$
          for all $out \in b$ **do**
            if $in \neq out$ **then** ▷ The entrance and exit should be different
              $\vec{s2} \leftarrow CalcStatus(in, out, current[gScore] + \vec{s1}[total\_time])$
              $\vec{s} \leftarrow \vec{s1} + \vec{s2}$
              $tentative\_gScore \leftarrow current[gScore] + \vec{s}[total\_time]$
              if $tentative\_gScore < b[gScore]$ and $checkConstraints(\theta, \vec{s})$ **then**
                $b[pPath] \leftarrow (current[exit], in, out, \vec{s})$
                $b[exit] \leftarrow out$
                $b[gScore] \leftarrow tentative\_gScore$
                $b[fScore] \leftarrow b[gScore] + h(b[exit], goal)$
                $b[camefrom] \leftarrow current$
                ▷ If b is already enqueued, replace it based on the b's state
                Enqueue($OPEN, b$)
              end if
            end if
          end for
        end for
      end if
    end for
  end if
end while

ALGORITHM 3: - Path Reconstruction Function Used in Algorithm 2

Input: *goal*: outdoor vertex;
Output: Recommended path p;

```
1: function ReconstructPath(goal)
2:     p ← ∅                                   ▷ Create an empty list
3:     current ← goal                          ▷ Set the current node with the goal outdoor vertex
4:     while current ≠ ∅ do
5:         p ← current ∪ p
6:         current ← current[cameFrom]
7:     end while
8:     return p
9: end function
```

During the iterations, the status $\vec{s1}$ to reach the next building b and the status $\vec{s2}$ of the indoor path in b are computed using the `CalcStatus` function as described above (lines 16 and 19). The status $\vec{s}$ of b is the integration of $\vec{s1}$ and $\vec{s2}$ and is a component of the state of b along with the partial path ($pPath$), exit door ($exit$), $gScore$, $fScore$, and the $current$ originating building ($cameFrom$)(set in lines 23–27).

A tentative $gScore$ is calculated using $current$'s $gScore$ and the total time t of the previously computed status s (line 21). If the tentative $gScore$ is less than b's $gScore$ and all the constraints θ are satisfied, then the $fScore$ is calculated based on a heuristic function h, and b is enqueued in the $OPEN$ priority queue. The $OPEN$ priority queue prioritizes the nodes based on the $fScore$. In case the node exists in the priority queue and its $gScore$ is greater than the node to be inserted, then the already enqueued node is replaced (line 28).

The `ReconstructPath` function shown in Algorithm 3 creates the found path using the $cameFrom$ attributes and the $pPath$ of each one of the nodes.

3 CONGESTION FORECASTING

The goal of congestion forecasting is to predict the future flow of congestion in a hallway or corridor in a given building at a given time. *ASTRO* only requires the congestion density for each segment of the building's indoor path (e.g., hallways or corridors) to ensure that the congestion tolerance constraint is satisfied. In this section, we discuss the two components of the congestion forecasting used by *ASTRO* (`PredictCongestion()` in Algorithm 1). First, we explain how the building grid layout is constructed (Section 3.1), and then we introduce the methodology that we employ for training and accessing the congestion models (Section 3.2).

3.1 Building Grid Layout

To improve the accuracy and efficiency of our congestion forecasting models, we divide the building corridors into smaller segments and develop a *Grid Layout* algorithm. The benefit of this approach is two-fold: (i) each segment can be trained and maintained separately, which allows us to massively parallelize the training process while forecasting at a fine-grained level; and (ii) it improves efficiency at inference time, since we only forecast the congestion at the cells of interest and not for the entire building.

Given the floor plan in a structured format (e.g., GeoJSON), we develop a new *Grid Layout* algorithm that overlays a uniform grid over the building and detects those cells that overlap with corridors, doors, and rooms. The output is a partial grid, the *Building Grid Layout*, that contains

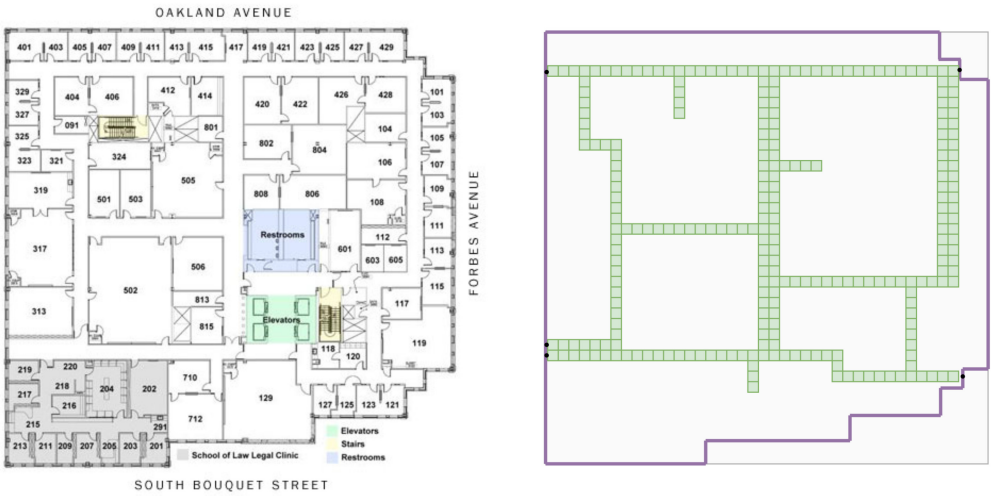

Fig. 4. Floor plan of the Sennott Square building at the University of Pittsburgh campus (left) converted to a Building Grid Layout (right).

only the cells that represent corridor segments, i.e., only the cells we are interested in modeling. The *Grid Layout* algorithm works as follows: First, the floor plan of the building is spatially indexed, i.e., corridors (polygons), rooms (polygons), and door locations (points) are loaded into a spatial index, specifically an R-tree. A uniform grid that covers the entire building is generated with a configurable cell size ϕ. Next, the grid cells are loaded into a second R-tree. Finally, a join operation between the floor plan components and the grid cells is performed using the previously constructed spatial indexes. Cells that overlap with any floor plan components, i.e., corridors, rooms, doors, are annotated as such whereas all other cells are discarded.

The output is a set of annotated grid cells. Figure 4 illustrates an example of such an output. On the left, the CS Department floor plan in the Sennot Square building at the University of Pittsburgh is depicted. The corresponding generated grid layout is shown on the right. The grid's boundary is displayed in the background as a gray rectangle, whereas the actual shape of the building is shown as a thick purple line. Building doors are marked with black dots and cells that correspond to corridor segments are colored. Other types of cells are omitted for brevity.

3.2 Congestion Prediction & Accessibility

Formally, we denote the congestion at a cell c as a signal $X_c \in \mathbb{R}^P$, where P is the number of features for the cell. Let $X_c^{(t)}$ be the congestion observed at cell c at timestep t. The congestion forecasting problem aims to learn a function $h(\cdot)$ that maps T' historical congestion signals to T future congestion signals:

$$\left[X_c^{(t-T'+1)}, \ldots, X_c^{(t)}\right] \xrightarrow{h(\cdot)} \left[X_c^{(t+1)}, \ldots, X_c^{(t+T)}\right]. \tag{2}$$

3.2.1 Training Phase. After the grid layout has been constructed, the training phase begins. For every corridor cell in the grid, a separate model is trained on the congestion data that correspond to that cell. To model the temporal dependencies of the congestion, we leverage ***Recurrent Neural Networks* (RNNs)**. For the multiple steps ahead forecasting, we employ the *Sequence-to-Sequence* architecture [46]. Both the encoder and decoder are recurrent neural networks and as we show in

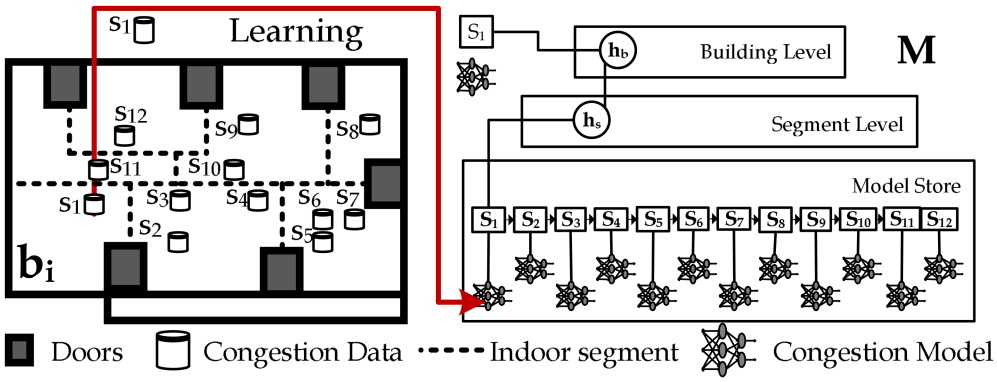

Fig. 5. The *CM-Structure* provides the *ASTRO* algorithm with improved performance. One forecasting model is created for each segment in the building based on its historical data.

our experiments (Section 5), we tested models with both LSTM and GRU neuron units as well as traditional fully connected RNN neurons.

During training, we feed the historical congestion time series into the encoder and use its final states to initialize the decoder. The decoder generates predictions given previous ground truth observations. At testing time, ground truth observations are replaced by predictions generated by the model itself. The entire network is trained by maximizing the likelihood of generating the target future time series using backpropagation through time.

3.2.2 Deployment and Inference Phase. As discussed earlier, at inference time the system has to be able to quickly and efficiently load the models of interest, i.e., the models that overlap with the indoor pathway, and make predictions. To this end, we construct a spatial index, namely, *CM-Structure*, that maps cells to forecasting models. The Congestion Forecasting component maintains the *CM-Structure*, shown in Figure 5, which stores the forecasting model of each corridor cell and provides a fast *Access Path*. This allows *ASTRO* to efficiently look up the models it requires while searching for the optimal path to the destination.

The *CM-Structure* Construction Algorithm is executed in an offline phase where the historical data are utilized to initialize and train a forecasting model for each corridor cell of the building model. Each model is inserted into the *CM-Structure* for faster retrievals during the last step of Algorithm 1. For example, the congestion data records of corridor cell c inside building b_i are used to train a model. Then, the trained model is inserted into the *CM-Structure* and becomes available to subsequent queries for any path P that intersects cell c.

3.2.3 Continuous Learning Phase. The dynamic nature of congestion can very often render the deployed congestion forecasting models obsolete. As the congestion dynamics change (e.g., the class schedule updated leading to more (or less) scheduled traffic or new cafeteria opened in the building) models need to keep adjusting to those changes [14]. To achieve that, we propose an offline periodic phase where existing congestion models are further trained on new congestion data collected since the last time the model was fine-tuned.

3.3 Data Collection

Obtaining indoor congestion datasets can be done in a variety of ways. Perhaps the most common way to collect indoor congestion data is through the analysis of Wi-Fi access point logs. In particular, given the number of connected users to an access point, the signal strengths of the established

connections, and the range of an access point, the expected number of pedestrians at different distance intervals can be computed. A more accurate method is through beacons. These devices have a shorter range and can therefore localize a device indoors with higher accuracy. Last, a recent work processed video streams in real-time to count the number of pedestrians passing from specific corridors. Unfortunately, such datasets are rarely released to the public and, therefore, in the next section, we propose a configurable method for generating realistic indoor congestion datasets.

4 INDOOR CONGESTION DATA GENERATOR

We design and implement a synthetic data generator due to the limited availability of indoor congestion data. If the synthetic data is generated for an existing building floor plan, then we utilize our developed *Building Grid Layout* algorithm. Otherwise, we first generate realistic building floor plans before we generate the congestion of a building using parameters that we obtained from real congestion scenarios. The generator contains two separate components, namely, the *Building Modeling* and the *Congestion Generator*, that can be executed independently of each other. Sections 4.1 and 4.2 describe the internals of each component in detail, respectively.

4.1 Building Modeling

Constructing a realistic building model is a crucial prerequisite for the next step, i.e., the synthetic congestion generator. The algorithm, which is an extension of *Building Grid Layout*, requires two inputs: the location of the building's doors and the grid cell size ϕ. For our experiments in Section 5, we set $\phi = 3$ meters.

Step 1 (Grid Construction): Initially, the shape of the building is approximated using the convex hull of the door locations. Note that an accurate representation of the building's shape is not required to generate a synthetic building model. The **minimum bounding rectangle (MBR)** of the doors (or of the convex hull) is used as the initial boundary of the grid. Subsequently, our algorithm generates the grid cells. The first cell is placed at the bottom left of the grid's initial boundary and cells are generated until the boundary is filled. The cells on the right and top edges may extend over the boundary and, hence, we update the grid boundary to reflect this. Cells that do not overlap with the building's shape are considered irrelevant and thus discarded.

Step 2 (Corridor Construction): Subsequently, the algorithm proceeds to generate corridors that connect all doors with each other. Our corridor generation algorithm has two objectives: (i) to minimize the number of direction changes, i.e., corridors must switch from horizontal to vertical orientation as few times as possible; and (ii) to reuse previous corridor segments whenever possible, as an architect would do in a real scenario. Satisfying both objectives leads to compact floor plans that are not flooded with corridor cells.

Step 3 (Room Placement): Last, a number of rooms is randomly placed along the unused cells of the grid and the corridors are extended to reach the doors of those rooms. The output is a set of grid cells, each annotated as CORRIDOR, ROOM, or UNUSED.

Figure 6 shows two examples of generated building models in which doors are marked as black dots, the grid's initial boundary with a light gray rectangle, corridor and room cells with green (dark gray in grayscale) and orange (light gray with texture in grayscale) squares, respectively. Room doors are depicted with brown textured squares (dark gray with texture in grayscale). Cells that remain unused after modeling are painted red (light gray in grayscale). Note that rooms are assumed to only have one entrance and therefore can only be used either as a source or as a destination, i.e., no path cuts through a room.

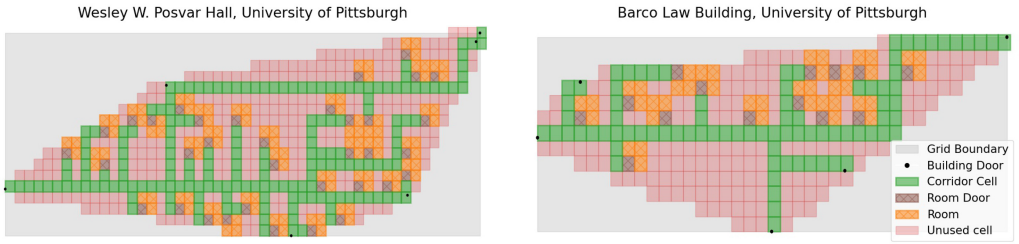

Fig. 6. Generated building models for a building with 6 doors and 35 rooms (left) and a building with 5 doors and 14 rooms (right) using $\phi = 3$ meters.

4.2 Congestion Generator

Our real-world observations show that traffic inside academic buildings can be separated into two main categories: *scheduled* and *pass-through*. Scheduled traffic is the traffic that follows a specific schedule, e.g., class schedule. This means that slightly before and slightly after a class, a spike of traffic appears in the building. Pass-through traffic is generated by people who enter the building in one door only to exit at another. Therefore, we design and implement a congestion generator that accounts for these two categories of traffic. To achieve this, congestion is generated in two phases. The first phase generates the pass-through traffic (browsers), whereas the second phase generates the scheduled traffic (commuters). Both phases require the building model as input as well as the date range of simulation. It is important to note that the building input does not necessarily have to come from the synthetic building modeler we introduced in the previous section. This means that the generator can simulate traffic for any building given its model.

Phase 1 (Pass-through traffic): The rate of congestion generated by this type of traffic varies, depending on the time of day. For example, there is almost zero traffic during nighttime but a lot of people will pass through the building during the daytime. Hence, the generator receives an additional input for this phase, namely, the arrival rate of people at different times of the day. For every timestep, the generator will randomly sample the number of people that arrive at each building door and for each person a destination exit door is selected. Then, assuming a constant walking speed, the trajectory of each person is simulated following the shortest path that is formed using corridor cells. The congestion of each cell is updated accordingly. It is important to highlight that, in this phase, the source and the destination of the simulated trajectories is always a building entrance/exit and that a trajectory will never involve a room or a room door, i.e., a simulated trajectory will never cut through a room.

Phase 2 (Scheduled traffic): Before beginning the scheduled traffic simulation, we need to have some sort of a schedule for every room in the building, i.e., the time classes start and end for different days of the week as well as the audience size of each class. Hence, the generator receives this schedule as an input for this phase. Obviously, a schedule does not exist for synthetic building models, so we will discuss the approach we employ to generate such schedules later on in this section. Besides the schedule, the generator requires an additional set of parameters, i.e., arrival rate of students before the class begins, departure rate after the class ends, and probability that a student is absent from class.

We simulate the arrival of students as a Gaussian distribution with mean μ_a and standard deviation σ_a that samples the number of minutes before the start time of the class the student enters the building. Similarly, we simulate the departure of students from class as a Gaussian distribution with mean μ_d and standard deviation σ_d that samples the number of minutes after the end time of

the class when the students leave the classroom. Last, we simulate the absence of a student from class as a binomial distribution with $p = \alpha$, An average walking speed v is assumed.

All parameters, i.e., $\mu_a, \sigma_a, \mu_d, \sigma_d, \alpha, v$, are user-defined. The scheduled traffic generator algorithm iterates over every schedule item and generates the respective traffic. For every student in the class, we first draw a sample from the absence distribution to decide if the student is attending the class. If the student is attending, then we additionally draw a sample from the arrival distribution to decide how early that student is going to enter the building. An entrance door is selected uniformly at random and the student's trajectory to the respective room is generated using a constant speed of v m/s. The congestion of the cells that intersect with the student's path is updated accordingly. A similar process is employed for the departure of students from the class. For every student that was present, a departure time and an exit door are sampled and the student is routed while the congestion of cells that intersect with the path is updated. In sum, for all arrival (departure) trajectories, the source is a building entrance (or room door) and the destination is a room door (building exit). The final output is the congestion time series of every corridor cell aggregated in five-minute intervals.

Discussion: Although we discuss (and make use of) the proposed generator in the context of academic buildings, the process can be generalized to almost any kind of building. The only hard requirement is the floor plan of the building. All other parameters ($\mu_a, \sigma_a, \mu_d, \sigma_d, \alpha, v$) can be provided by the user or estimated using other datasets, such as Wi-Fi access point connection logs. For example, at USC, access point logs are analyzed to estimate and predict the congestion in buildings during the pandemic to reduce the spread of the virus. The logs tend to show a spike in the number of connections a few minutes before the start of a class and a drop in the number of connected users right after a class finishes. These spikes and drops can be correlated with the schedule to estimate the parameters $\mu_a, \sigma_a, \mu_d, \sigma_d$.

Schedule Generator: In the scenario that a schedule does not exist for a building, we propose a simple method to generate one. Three input parameters are required: (i) the audience size distribution, which is assumed to be a Gaussian distribution with mean μ_s and standard deviation σ_s; (ii) a set of event durations (class, no class) along with their respective probabilities, which are modeled as multinomial distribution; and (iii) the probability p_c that an event is a class or not. Optionally, the time of day when classes begin and end, e.g., from 7AM to 9PM, can be provided. Starting at the time that the schedule is configured to begin, e.g., 7AM, we sample two pieces of information, namely, the duration of the next event δ and whether the next event is a class or not. If the next event is a class, then the size of the audience is also sampled.

Figure 7 plots the generated congestion of two corridor cells during the first week of April 2019. On the top, the congestion of a corridor cell at the building's entrance is shown while on the bottom the congestion of an intersection cell is shown. Both cells follow a realistic distribution in the sense that during the nighttime only a very small number of pedestrians is observed, whereas the number of pedestrians increases during the day and peaks around the beginning and ending of classes. As expected, the congestion at the intersection is much higher than at the entrance. This happens because pedestrians are very likely to walk by the intersection irrespective of the entrance or exit they used.

5 EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of our proposed methods. We start out with the experimental methodology and setup, followed by three experiments. The first experiment (Section 5.2) examines the influence of machine learning models described in Section 3. The second experiment (Section 5.3) investigates the tradeoffs of three *CM-Structure* implementations. Finally,

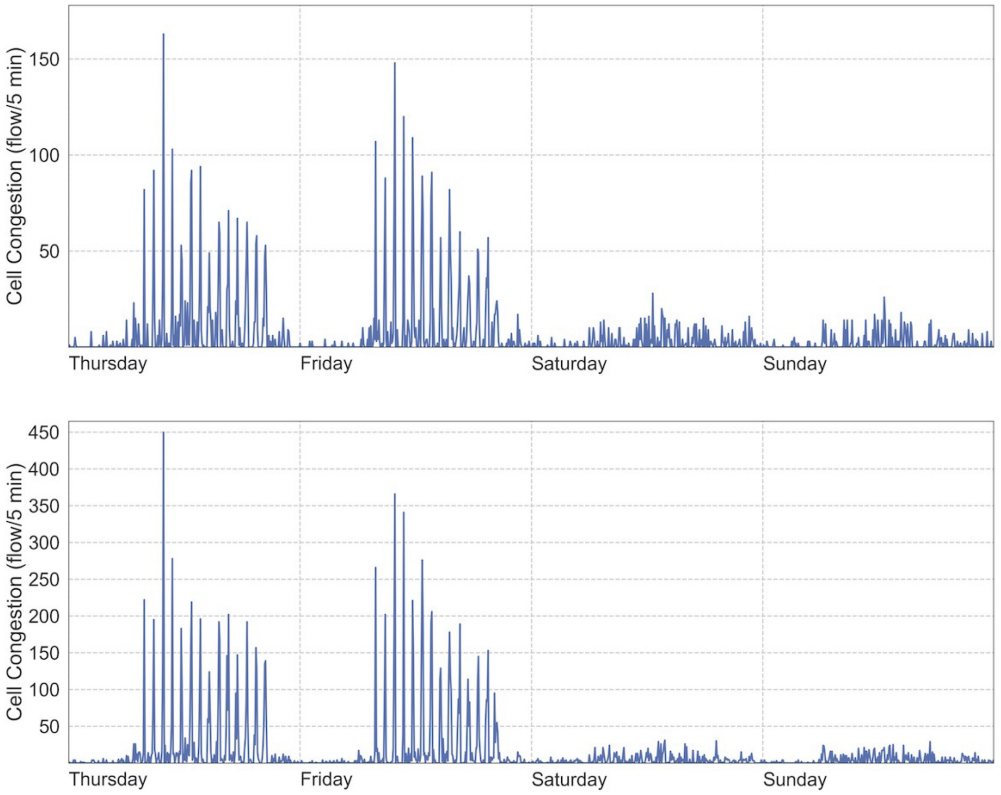

Fig. 7. Generated congestion for a corridor cell corresponding to Entrance 1 (top) and a corridor intersection (bottom) of Wesley W. Posvar Hall at University of Pittsburgh.

in the third experiment (Section 5.4), the performance of *ASTRO* is compared against three baseline approaches with respect to the quality of recommendation and the cost of execution.

5.1 Experimental Setup

This section provides details regarding the algorithms, metrics, and datasets used for evaluating the performance of the proposed approach.

Testbed: Our evaluation is carried out on a dedicated Windows 10 server. The server is featuring 12 GB of RAM with 4 Cores (@ 2.90 GHz), a 500 GB SSD and a 750 GB HDD. We implement all path search algorithms using Scala 2.13 and the three machine learning models (FC-RNN, GRU, LSTM) utilizing Keras with Tensorflow [1] on an NVIDIA GeForce RTX 3080 GPU.

Algorithms: The proposed *ASTRO* algorithm (see Algorithm 2) is compared with the following approaches:

- **Dijkstra:** This is a modified and optimized version of Dijkstra using a priority queue and early termination with additional constraints similar to *ASTRO*.
- **ASTROG:** This is a greedy variant of *ASTRO* where the indoor path is selected based on the minimum indoor time instead of the overall time or the heuristic.

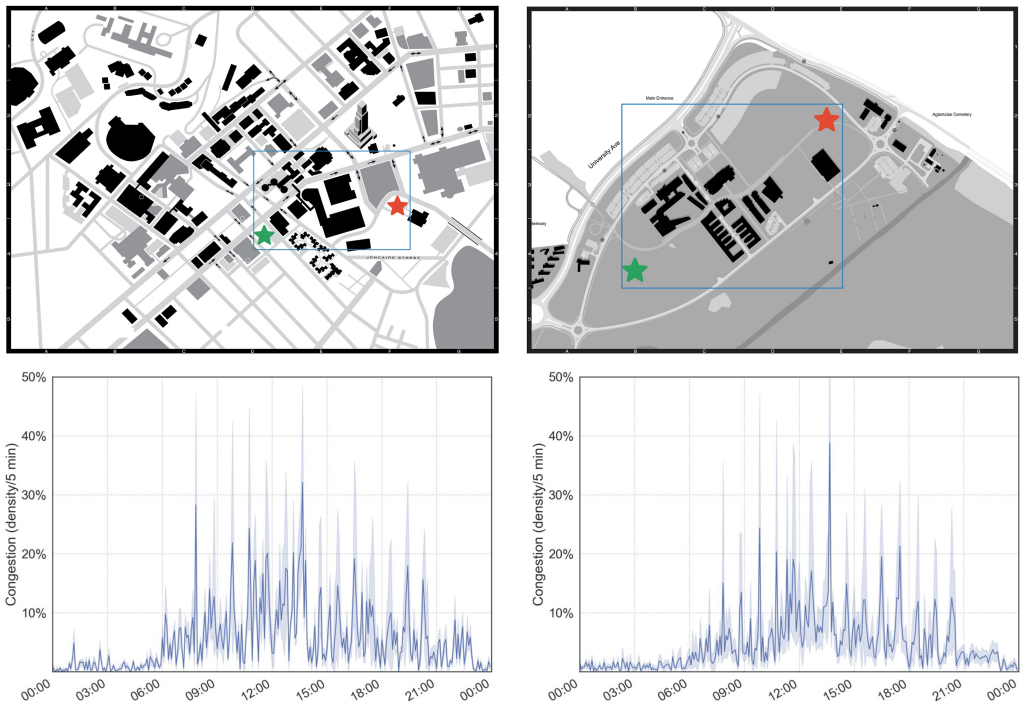

Fig. 8. Datasets for the buildings in the University of Pittsburgh campus (top left) and the University of Cyprus campus (top right) and their average congestion for one day (bottom row). The green and red stars represent the source and destination nodes used in the experimental evaluation, and the blue box the search space for all algorithms.

- **CBFS (Closest Building First Search):** This is an algorithm where the closest building is always selected first.

Note that *ASTRO* is the core algorithm of the proposed solution in this article, Algorithm 2.

Datasets: We utilize the following two datasets in our evaluation:

- **PITT**: A realistic dataset of the University of Pittsburgh campus. It consists of 9 buildings with each building having 2 to 6 doors (3 on average) and up to 582 corridor cells (126 on average). The average door-to-door corridor length is 69 meters.
- **UCY**: A realistic dataset of the University of Cyprus campus. It consists of 9 buildings with each building having 2 to 7 doors (4 on average) and up to 396 corridor cells (106 on average). The average door-to-door corridor length is 48.5 meters.

We used camera analysis [15] on a two-hour session and extrapolated the congestion data using the University of Pittsburgh Fall 2019 schedule. Then, for both *PITT* and *UCY* datasets, we generate congestion data using the procedure we described in Section 4 using $\mu_a = 4, \sigma_a = 1, \mu_d = 1, \sigma_d = 0.2$ and a walking speed of 1.4 m/s [6, 24] for a period of six months. Figure 8 shows the map of the two campuses and the average generated congestion density (number of people over corridor capacity) of all buildings on July 19, 2019.

Metrics: We evaluate the performance of *ASTRO* using the following metrics:

- **Outdoor exposure (*O*):** measures the total outdoor exposure of the recommended path in seconds.

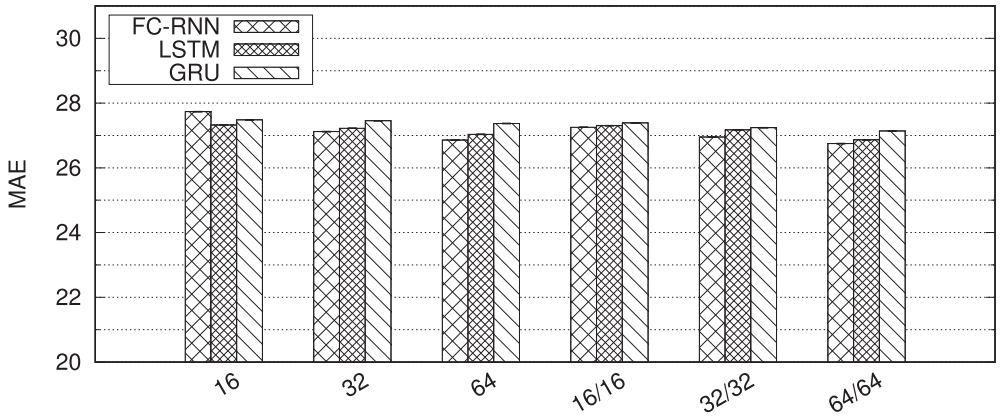

Fig. 9. Examining the effect of hidden layers and number of units on the accuracy of the congestion models for Wesley W. Posvar Hall at University of Pittsburgh.

- **Indoor Time (*I*):** measures the total indoor time of the generated path in seconds.
- **Total Time (*T*):** measures the total travel time of the path between the source and the destination point in seconds.
- **Congestion Reduction (*C*):** measures the indoor congestion of the generated path.
- **Response Time:** measures the execution time using the average of 100 consecutive runs in milliseconds.
- **Memory (*M*):** measures the memory footprint of the ML model in kilobytes.
- **Root Mean Squared Error (*RMSE*):** measures the error of the model and is computed using the formula $RMSE(X, \hat{X}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}$, where X is the ground truth and $\hat{X}$ the prediction.
- **Mean Absolute Error (*MAE*):** measures the error of the model and is computed using the formula $MAE(X, \hat{X}) = \frac{1}{N}\sum_{i=1}^{N}|X - \hat{X}|$.
- **Query Throughput:** measures the maximum number of queries that the system can sustain within a second in **Queries Per Second (*QPS*)**.

5.2 Experiment 1: Congestion Forecasting Evaluation

We conduct our experiments on two buildings of the PITT dataset, namely, ***Wesley W. Posvar Hall* (*posvar*)** and ***Barco Law Building* (*barco*)**. 70% of the data is used for training, 10% for validation, and the remainder 20% for testing. Additionally, we apply standard normalization as a pre-processing step to prepare the datasets for training and we include the time of day as a feature. We compare the performance of three time series regression models (seq-to-seq encoder-decoder architecture); (i) a **fully connected RNN model (FC-RNN)**, (ii) a **Gated Recurrent Unit RNN model (GRU)**, and (iii) a **Long-Short Term Memory RNN model (LSTM)**. All models were trained for 30 epochs using the Adam optimizer and a learning rate of $5E{-}3$ with a decay rate of 0.1 every 200 steps and with a batch size of 256. During training (and inference), we provide the last hour's congestion in 5-minute windows as input to the model and receive the next hour's congestion as output, i.e., 12 timesteps as input, 12 timesteps as output.

First, we train the models using *posvar*'s congestion data, which has the largest number of grid cells to tune them in terms of the number of layers and units hyperparameters. Specifically, we performed experiments using 1 hidden layer (with 16, 32, and 64 units) and 2 hidden layers (with 16/16, 32/32, and 64/64 units). Figure 9 plots the results. On the horizontal axis, we vary the number

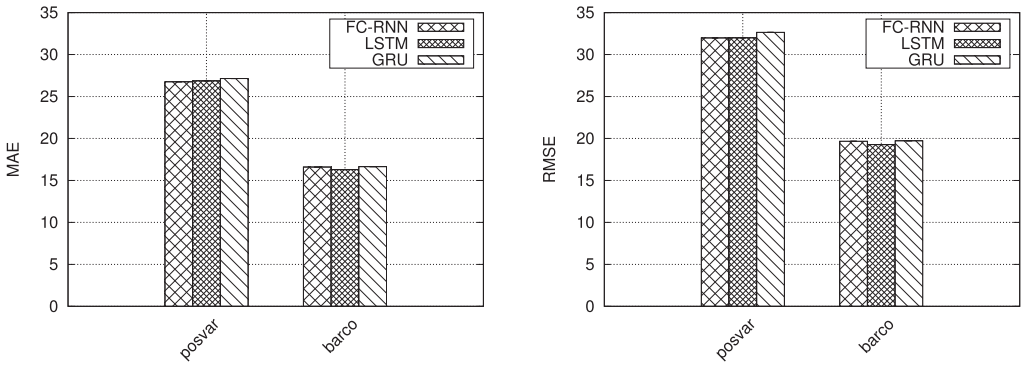

Fig. 10. Examining the *MAE* (left) and *RMSE* (right) of the three ML models (2 hidden layers, 64 units each) for *Wesley W. Posvar Hall* and *Barco Law* buildings at University of Pittsburgh.

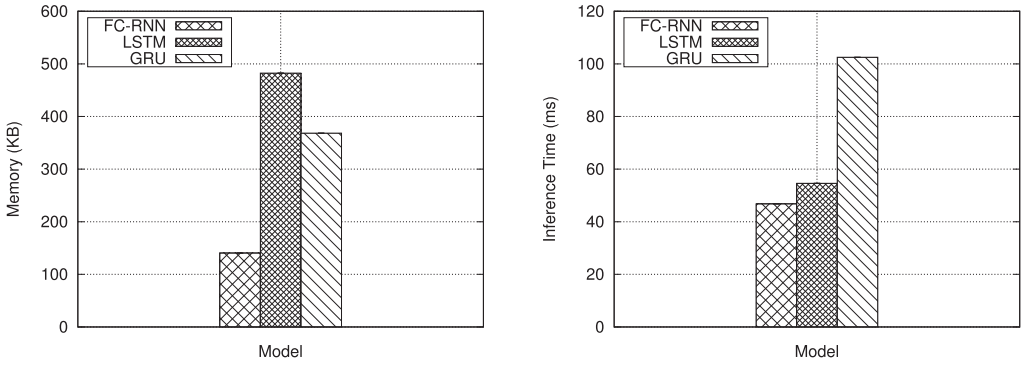

Fig. 11. Examining the memory footprint (left) and inference time (right) of the three ML models (2 hidden layers, 64 units each).

of hidden layers and number of units combination and on the vertical axis we plot the corresponding model's MAE. We observe that the smallest error is achieved for all models when they use 2 hidden layers with 64 units each. Hence, for the remainder of the experiments, all the models were trained using 2 hidden layers with 64 units each.

Next, we investigate three different aspects of the tuned models: (i) their accuracy, (ii) their memory footprint, and (iii) their inference time. It is crucial that our system adopts not only a model that is very accurate but also one that does not require a lot of memory and incurs small inference time, since, to predict the congestion of a path, tens of models, i.e., models corresponding to corridor cells that overlap with the path, may need to be invoked.

Figure 10 examines the accuracy of the tuned models. Specifically, the average *MAE* (left) and *RMSE* (right) metrics of all models (corresponding to grid cells) in each building is shown. We observe that all models are able to achieve similar accuracy for both buildings. Therefore, all models satisfy the requirement of high prediction accuracy and are suitable candidates for *ASTRO*.

Figure 11 (left) examines the memory footprint of the models. As expected, due to its larger number of parameters, LSTM requires more memory, making it a less ideal model to adopt by our proposed approach for the reasons explained earlier. FC-RNN, however, requires much less memory while achieving almost identical accuracy to the other models. Therefore, in terms of memory requirements, FC-RNN and GRU are better candidates for *ASTRO*.

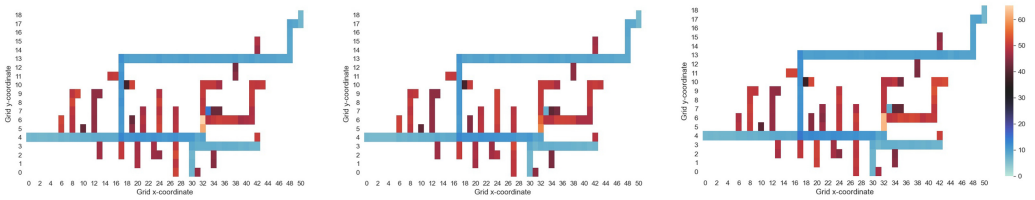

Fig. 12. Examining the error of every corridor cell in a building in terms of *MAE* for FC-RNN (left), LSTM (middle), and GRU (right) for Wesley W. Posvar Hall at University of Pittsburgh.

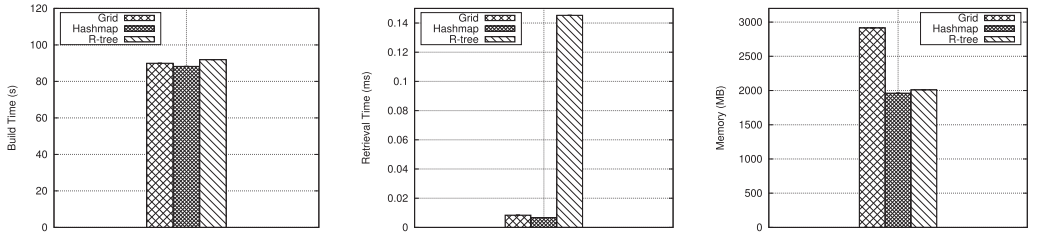

Fig. 13. Examining the performance of the Congestion Model Access Methods in terms of build time (left), retrieval time (center), and memory footprint (right).

Finally, Figure 11 (right) examines the inference time of the models. We observe that FC-RNN incurs the smallest inference time of 40 milliseconds per cell, while GRU incurs 2.5× more time than FC-RNN. This shows that only FC-RNN and LSTM satisfy the requirement of incurring small inference time.

Furthermore, we investigate which corridor cells incur the greatest error. Figure 12 examines the accuracy of every corridor cell in *posvar*, the largest building in the *PITT* dataset, in terms of *MAE*. This confirms our initial observations that all models perform similarly. It is also not surprising that corridors leading only to rooms and not exit doors, incur the highest error. The reason behind this is that, intuitively, congestion sporadically appears at those cells slightly before and exactly after a class. This kind of corridors only facilitate "commuters" and, hence, have almost 0 congestion most of the time, which makes it very difficult for the model to predict how congestion is going to change in the immediate future.

Summary: Taking all the above into consideration, we reach the conclusion that FC-RNN is the most ideal model to be adopted by *ASTRO*. This is due to the fact that it manages to satisfy all three requirements; (i) high accuracy; (ii) low memory footprint; and (iii) low inference time. Specifically, it achieves comparable accuracy to the other two models while requiring up to 4× less memory and incurring up to 2.5× less inference time.

5.3 Experiment 2: Congestion Model Access Method Evaluation

It is critical for our proposed algorithm to quickly identify and retrieve all models that intersect a corridor segment. Hence, we run experiments for three different *Access Method* implementations; (i) a Grid that stores models in a 2D array; (ii) a hashmap that maps cells to models; and (iii) an R-tree that indexes the cells based on their spatial boundaries. Figure 13 (left) illustrates the time required to load the models of *posvar* and construct the respective *Access Path* structure. All three structures incur relatively similar construction times. Figure 13 (center) examines the average time it takes to retrieve all the models that intersect a corridor. R-tree incurs the highest retrieval time, whereas the other two structures incur comparable times. Last, Figure 13 (right) investigates the

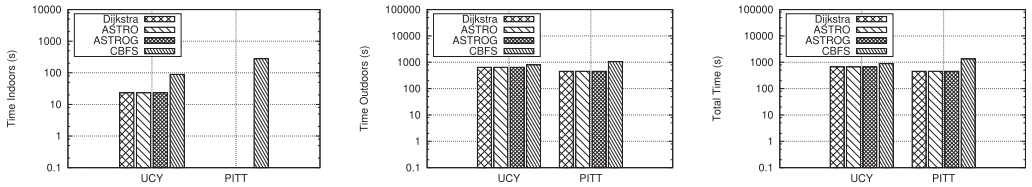

Fig. 14. Examining the indoor total time (left), the outdoor total time (center), and the total travel time (right) in seconds for finding the longest path in the PITT and UCY datasets without any constraints.

memory footprint of the structures. As expected, the Grid is the most expensive, since it allocates space for all cells in the grid even though only a portion of them corresponds to corridors, whereas the other two only index corridor cells.

Summary: Although both *Grid* and *Hashmap* Access Methods satisfy *ASTRO*'s requirements, we decide to adopt *Hashmap*, since it is slightly faster and requires 33% less memory than *Grid*.

5.4 Experiment 3: *ASTRO* Performance Evaluation

In the third set of experiments, we evaluate the performance of our proposed *ASTRO* algorithm against three baseline solutions using the datasets described in Section 5.1. We select source and destination nodes that maximize the search space for all algorithms, as shown in Figure 8. This allows the algorithms to consider paths that span multiple buildings and satisfy the constraints described in Section 2.1. We use Manhattan distance to better represent the pedestrian path and an average walking speed of 1.4 m/s [6, 24].

5.4.1 Series 1: Absence of Constraints. In the first set of experiments, we focus solely on optimizing the total travel time, i.e., setting the constraints to $A = false$, $OE = \infty$, and $CT = 1.0$ (100%). The algorithms do not take into account any constraints, but the delay caused by congestion is considered in the calculation of the total travel time.

Dijkstra, *ASTRO*, and *ASTROG* behave identically (returning identical paths), as shown in Figure 14. All three incur 0 indoor time for PITT dataset with the path being totally outdoors, due to the absence of the outdoor exposure limit constraint. Interestingly, all three incur the same indoor time for UCY dataset (23 seconds) by finding a path through a building that forms a shortcut to the destination. The total time of the path is 456 seconds for PITT and 670 seconds for UCY.

In contrast, *CBFS* spends more than 270 seconds indoors for PITT dataset and almost 90 seconds for UCY dataset incurring 10% congestion. This happens because its strategy is to visit the closest building first irrespective of the presence or absence of constraints, therefore increasing the total travel time as shown in Figure 14 (right). The total time of the path provided by *CBFS* is 3× larger than the path recommended by the other approaches.

In Figure 15, we can easily observe that *ASTRO*, *ASTROG*, and *CBFS* perform the same with *ASTRO* being slightly faster for PITT dataset. In contrast, *Dijkstra* incurs 100× and 10× more time (for UCY and PITT datasets, respectively) in finding the optimal path compared to *ASTRO*. This happens because *Dijkstra*'s search space is much larger, especially in the UCY dataset where the graph is very dense (i.e., the buildings are very close to each other). *Dijkstra*, *ASTRO*, *ASTROG*, and *CBFS* report 0 average and maximum congestion. However, *CBFS* has an average indoor congestion of approximately 10% and a maximum indoor congestion of 100%.

Dijkstra and *ASTRO* algorithms are optimal. This means that both algorithms find the path with minimum total travel time and at the same time satisfy all the constraints. However, *ASTRO* outperforms *Dijkstra* by up to 13× in terms of execution time.

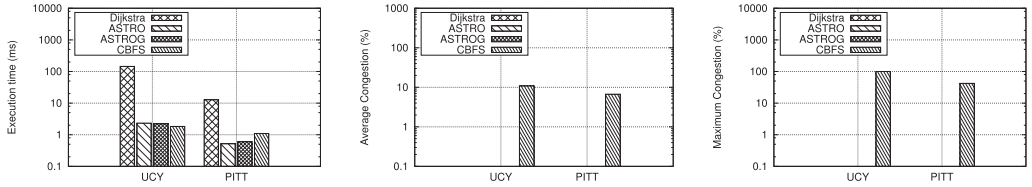

Fig. 15. Examining the response time (left) in milliseconds, the average congestion (center), and the maximum congestion (right) in percentage for finding the longest path in the PITT and UCY datasets without any constraints.

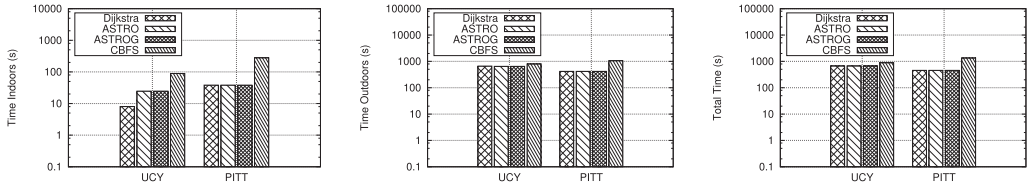

Fig. 16. Examining the indoor total time (left), the outdoor total time (center), and the total travel time (right) in seconds for finding the longest path in the PITT and UCY datasets with outdoor exposure time limit of 300 seconds.

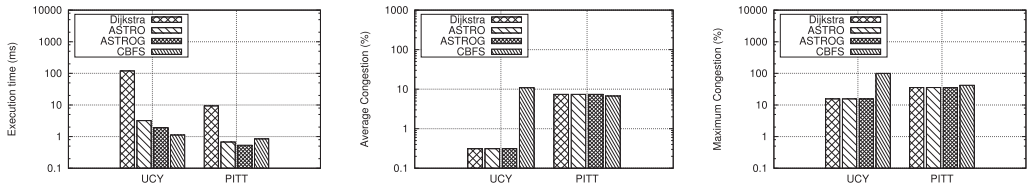

Fig. 17. Examining the response time (left) in milliseconds, the average congestion (center), and the maximum congestion (right) in percentage for finding the longest path in the PITT and UCY datasets with outdoor exposure time limit of 300 seconds.

5.4.2 Series 2: Outdoor Exposure Constraint (300 Seconds). In the second set of experiments, we focus on optimizing the total travel time while keeping the outdoor exposure time less than 300 seconds, i.e., setting the constraints to $A = false$, $OE = 300$, and $CT = 1.0$ (100%).

Figure 16 shows the effect of the 300-second outdoor exposure limit. Particularly, *Dijkstra* incurs 8 seconds of indoor time for UCY and 38 seconds for PITT. *ASTRO* and *ASTROG* behave identically, i.e., they recommend the exact same path, which incurs approximately 25 and 38 seconds of indoor time for UCY and PITT, respectively. *CBFS*, however, recommends the same paths as before for both datasets. The total time for both datasets remains the same as in the previous experiment.

In Figure 17, we observe that the response time for all of the approaches is slightly decreased in comparison to the previous experiment (shown in Figure 15). This happens because several edges are pruned due to the outdoor exposure constraint. However, *ASTRO*, *ASTROG*, and *Dijkstra* have the same average indoor congestion (7% for PITT and 0.3% for UCY). All three of them incur 35% maximum congestion for PITT and 16% for UCY. *CBFS* has the same average and maximum congestion with the previous experiment shown in Figure 15 (right).

5.4.3 Series 3: Outdoor Exposure Constraint (300 Seconds) & Congestion Tolerance Limit (15%). In the third set of experiments, we focus on optimizing the total travel time while keeping the

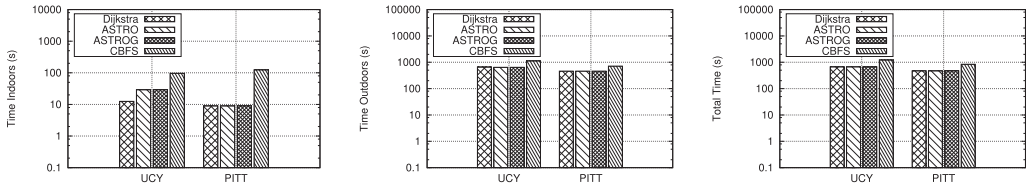

Fig. 18. Examining the indoor total time (left), the outdoor total time (center), and the total travel time (right) in seconds for finding the longest path in the PITT and UCY datasets with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit.

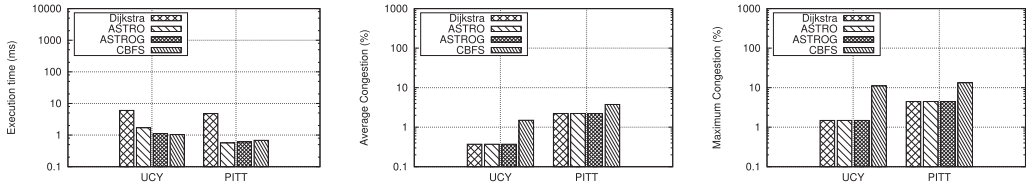

Fig. 19. Examining the response time (left) in milliseconds, the average congestion (center), and the maximum congestion (right) in percentage for finding the longest path in the PITT and UCY datasets with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit.

outdoor exposure time less than 300 seconds, and the congestion tolerance limit less than 15%, i.e., setting the constraints to $A = false$, $OE = 300$, and $CT = 0.15$ (15%).

Figure 18 shows the effect of 300 seconds outdoor exposure limit and 15% congestion tolerance limit. Specifically, *Dijkstra* incurs approximately 9 seconds of indoor time for PITT and 12 seconds for UCY. *ASTRO* and *ASTROG* behave identically recommending again the exact same path, which incurs approximately 9 seconds of indoor time for PITT and 29 seconds for UCY. *Dijkstra*, *ASTRO,* and *ASTROG* have the same total travel time of 670 seconds for UCY and 470 seconds for PITT (14 seconds overhead from the previous experiment). *CBFS* incurs 97 seconds and 124 seconds for UCY and PITT datasets, respectively.

Figure 19 shows that the congestion constraint affects the response time because additional paths are pruned. Particularly, *Dijkstra*'s response time drops to 6 milliseconds from 120 milliseconds for UCY datset and to 4 milliseconds from 9 milliseconds for PITT dataset. Finally, the average indoor congestion is decreased for all approaches. *ASTRO*, *ASTROG*, and *Dijkstra* have the same average indoor congestion 2% for PITT dataset and 0.3% for UCY dataset. Their maximum congestion is 4% and 1% for UCY and PITT datasets, respectively. *CBFS* reports 4% for PITT and 2% for UCY, while the maximum congestion is 11% and 14% for UCY and PITT datasets, respectively.

Summary: Even though both *Dijkstra* and *ASTRO* guarantee finding the optimal path in terms of satisfying the constraints and minimizing the total travel time, *ASTRO* is approximately one order of magnitude faster when constraints are imposed and two orders of magnitude faster when they are not. *ASTRO* reduces the maximum congestion by one order of magnitude compared to algorithms that are oblivious to congestion while maintaining comparable response time with the greedy approaches.

5.4.4 Series 4: Practicality of Our Solution. In this set of experiments, we are examining the practicality of our solution by measuring the throughput for the longest path in PITT and UCY datasets.

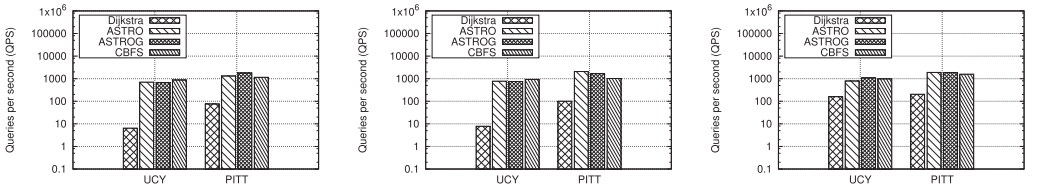

Fig. 20. Examining the throughput, in queries per second (QPS), without any constraints (left), with outdoor exposure time limit of 300 seconds (center), and with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit (right) for finding the longest path in the PITT and UCY datasets.

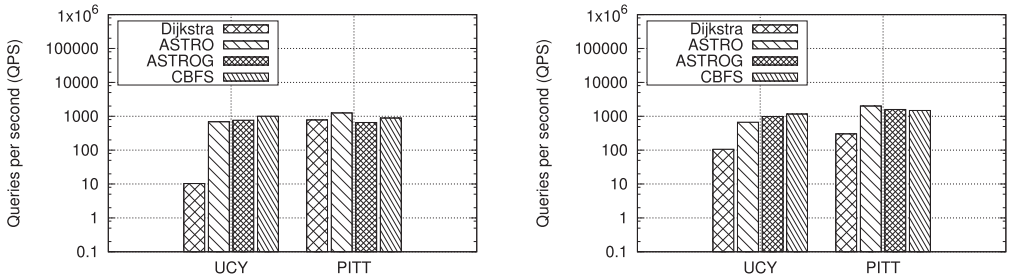

Fig. 21. Examining the throughput with outdoor exposure time limit ranging from 100 to 500 seconds (left), and with outdoor exposure time limit ranging from 100 to 500 seconds and congestion tolerance limit from 5% to 20% (right) in queries per second (QPS) for finding the longest path in the PITT and UCY datasets.

Figure 20 shows the throughput in **queries per second (QPS)** without any constraints, with outdoor exposure time limit of 300 seconds, and with outdoor exposure time limit of 300 seconds and 15% congestion tolerance limit for finding the longest path in the PITT and UCY datasets. *ASTRO*, *ASTROG*, and *CBFS* outperform *Dijkstra* by up to two orders of magnitude for UCY dataset and up to one order of magnitude for PITT dataset. Particularly, *Dijkstra* incurs 58 QPS for UCY dataset and 127 QPS for PITT dataset on average, *ASTRO* incurs 757 QPS for UCY dataset and 1,757 QPS for PITT dataset on average, *ASTROG* incurs 835 QPS for UCY dataset and 1,780 QPS for PITT dataset on average, and *CBFS* incurs 925 QPS for UCY dataset and 1,253 QPS for PITT dataset on average.

Figure 21 shows the throughput in QPS with outdoor exposure time limit ranging from 100 to 500 seconds and congestion tolerance limit from 5% to 20% for finding the longest path in PITT and UCY datasets. *ASTRO*, *ASTROG*, and *CBFS* outperform *Dijkstra* by up to 2× on average for both experiments for the PITT dataset and up to one order of magnitude on average for UCY dataset.

Summary: Even though both *Dijkstra* and *ASTRO* guarantee finding the optimal path in terms of satisfying the constraints and minimizing the total travel time, *ASTRO* is up to two orders of magnitude faster in most cases. This set of experiments shows that *ASTRO*'s high throughput supports route navigation incorporating congestion updates in real time. In contrast, traditional graph-based algorithms cannot cope with the requirements of modern navigation (e.g., Google Maps) and MCAN systems (e.g., HealthDist [11, 38]). In this setting, an initial route is recommended and then periodically refined to account for any traffic changes in real time.

6 RELATED WORK

In this section, we present the background and related work in systems that can provide outdoor, indoor, or combined navigation and path recommendation. We also include the related work for the traffic prediction for completeness.

6.1 Outdoor Path Recommendation

Outdoor systems are well established and enhanced through a variety of data collection and processing techniques, e.g., OpenStreetMap, Google Maps, Bing Maps, Here WeGo, TomTom, Waze. Several systems integrate social network data or crowdsourcing data and produce enriched path recommendation using machine learning and providing alternative navigation services using immersive technologies [7]. For example, *Dejavu* is a path navigation system that utilizes cell-phone sensors to provide accurate and energy-efficient outdoor localization [5]. Gervey et al. demonstrate how an alternative outdoor path can be generated based on the safety of a route [19]. However, there are systems that are trying to provide the fastest and simplest route for a destination [43]. Mata et al. show how social network data from a user profile may affect the outdoor path recommendation [36] and enrich it using augmented reality navigation [35]. Other systems focus on the temporal or personal preferences of the user to discover outdoor activities [40].

6.2 Indoor Path Recommendation

With people spending the 90% of their time indoors [53], indoor navigation services optimized the shortest path search using magnetic fields for localization and a modified shortest path formulation [50]. An indoor environment has many elements with unique properties that are defining the indoor route [44]. Additionally, marking the indoor environment (e.g., Braille blocks) can assist the visual impaired people to navigate indoors [42].

Smart home and wireless sensor networks provide the opportunity to localize and navigate indoors with spatial awareness [26]. Indoor localization and navigation services have emerged due to the rapid growth of new large buildings (e.g., shopping centers, campuses, building complexes).

Anyplace is an infrastructure-free indoor navigation system that uses sensing data from smartphones to determine the user's location [53]. Delail et al. proposed a context-aware system that enriches the indoor information by using augmented reality to provide indoor navigation [12].

Indoor environment and context are very important for determining the best indoor navigation route, especially to impaired people [17, 30]. Park et al. propose an indoor pedestrian network data model for emergency transportation services [39]. Afyouni et al. illustrate how to process indoor continuous path queries over traditional database management systems using a hierarchical, context-aware data model [3]. Feng et al. propose keyword-aware indoor path recommendation [16]. Last, Liu et al. propose the temporal variations-aware indoor path planning that takes the operating hours of shops into consideration [33]. However, none of the previous works consider dynamic edge weights such as the congestion, which is of critical importance during a pandemic.

6.3 Indoor-outdoor Path Recommendation

The majority of the indoor-outdoor systems focus on the seamless transition between indoor and outdoor navigation [34, 45]. Additionally, *IODetector* detects the indoor-outdoor environment changes accurately and efficiently, allowing the development of context-aware mobile applications [31]. *IONavi* is a joint indoor-outdoor navigation solution that uses mobile crowdsensing to create a collection of indoor-outdoor paths and then produces an indoor-outdoor path based on the generated collection [48]. Similarly, *CrowdNavi* solves the last-mile navigation problem using crowdsourcing and the guider-follower model [49]. Jensen et al. presented a unified model of indoor and outdoor spaces that can provide the shortest path by exploiting the nature of buildings and roads [23]. CAPRIO [8–10] is the latest indoor-outdoor path planning system that supports shortest path recommendations. CAPRIO users can specify outdoor exposure tolerance similar to *ASTRO*. However, these solutions are multi-objective path recommendation systems that cannot cope with multiple constraints and do not operate over time-varying graphs.

6.4 Constraint-based Path Selection

Previous work on constraint-based path selection was mostly focused on QoS routing, and their task is to find *feasible* paths while providing some guarantees. In Reference [25] the authors propose a **multi-constrained optimal path (MCOP)** selection algorithm that can be applied to any number of constraints. The authors of References [27, 28] provide a complete overview of all the proposed solutions. However, these solutions cannot be directly applied to ASTRO. These algorithms attempt to either find any path that satisfies the constraints or the optimal path in terms of a nonlinear cost function that involves the constraints over a single graph. Thus, their path-finding approaches, which are focused on QoS, cannot seamlessly integrate outdoor and indoor graphs to recommend routes for pedestrians.

6.5 Congestion Forecasting & Predictive Path Planning

Congestion forecasting is a classic problem in intelligent transportation systems research where the task is, given the current traffic situation, to forecast the future traffic situation. Traditionally, time-series regression techniques such as ARIMA, VAR, and others have been applied to predict the future situation of the traffic. Recently, more advanced techniques that involve complex and deep neural networks have been studied and are now considered state-of-the-art [32]. Additionally, some traffic prediction solutions, besides road monitoring data, also incorporate online information about events and weather conditions to achieve higher accuracy [55]. However, our work focuses on indoor foot traffic congestion, a task that is entirely different in the sense that conditions can change very fast; from little congestion in one minute, to high congestion in the next minute, and back to little congestion. Furthermore, in the transportation domain, the task is to forecast the congestion at a specific road segment (or sensor) that is strictly uni-directional, whereas corridors are bidirectional (people can walk in opposite directions at the same time at the same corridor).

Considering only the current traffic and assuming that it remains constant thereafter leads to suboptimal results due to the highly dynamic nature of indoor congestion. Therefore, it is imperative that *predictive path planning* [13] is employed to discover the best possible path that satisfies the constraints by predicting how congested a corridor will be when the person arrives there.

7 CONCLUSIONS AND FUTURE WORK

In this article, we present *ASTRO*, a novel graph-based path discovering algorithm that reduces the risk of COVID-19 exposure and enables ***Mobile Contact Avoidance Navigation (MCAN)***. *ASTRO* algorithm integrates outdoor nodes (i.e., buildings) with indoor nodes (e.g., doors, stairs, escalators, elevators) to efficiently provide a personalized contact-avoidance path satisfying a user-specified set of constraints (i.e., accessibility requirements, congestion tolerance, arrival time, and outdoor exposure). *ASTRO* uses a two-level data structure, dubbed *CM-Structure*, which allows the algorithm to quickly retrieve the congestion models that are relevant to a corridor to forecast the congestion at a given point in time. A second contribution of this article is an experimental platform for indoor congestion generation to support congestion forecasting in indoor spaces. We have developed this platform as part of *ASTRO*'s evaluation to address the lack of readily available indoor congestion data to train ML models. Our experimental results show that *ASTRO* can reduce the risk of COVID-19 exposure by up to one order of magnitude while maintaining similar response time compared to algorithms that are oblivious to congestion.

Beyond MCAN, *ASTRO* can be used as a core component in the development of evacuation algorithms [47] and group recommendations where multiple requests arrived/submitted need to be considered together. In the future, we plan to incorporate *ASTRO* and *CM-Structure* in

HealthDist [11, 38], our first MCAN prototype system. *HealthDist* can provide accessibility information about individuals for more accurate walking speed, which can be used by *ASTRO* to enhance prediction accuracy. We also aim to enhance the congestion modeling process to further improve the forecasting accuracy and efficiency of the models by exploring new machine learning models that exploit state and real-time information from crowdsourcing and sensing devices. Last, we also plan to investigate novel path scheduling solutions at the system level to prevent congestion hot-spots using participatory services.

ACKNOWLEDGMENTS

We thank Brian Nixon, member of our group at the University of Pittsburgh for the assistance with the implementation. We also thank Thalia Chrysanthis and Diomides Papadiomidous for their valuable help in proofreading the manuscript.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the sponsors.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, 265–283.

[2] Ahmed F. Abdelghany, Khaled F. Abdelghany, Hani S. Mahmassani, and Wael Alhalabi. 2014. Modeling framework for optimal evacuation of large-scale crowded pedestrian facilities. *Eur. J. Oper. Res.* 237, 3 (2014), 1105–1118. DOI: https://doi.org/10.1016/j.ejor.2014.02.054

[3] Imad Afyouni, Cyril Ray, Sergio Ilarri, and Christophe Claramunt. 2014. A PostgreSQL extension for continuous path and range queries in indoor mobile environments. *Pervas. Mob. Comput.* 15 (2014), 128–150. DOI: https://doi.org/10.1016/j.pmcj.2013.09.008

[4] Nadeem Ahmed, Regio A. Michelin, Wanli Xue, Sushmita Ruj, Robert A. Malaney, Salil S. Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K. Jha. 2020. A survey of COVID-19 contact tracing apps. *IEEE Access* 8 (2020), 134577–134601. DOI: https://doi.org/10.1109/ACCESS.2020.3010226

[5] Heba Aly, Anas Basalamah, and Moustafa Youssef. 2017. Accurate and energy-efficient GPS-Less outdoor localization. *ACM Trans. Spatial Algor. Syst.* 3, 2 (2017), 4:1–4:31. DOI: https://doi.org/10.1145/3085575

[6] Richard W. Bohannon and A. Williams Andrews. 2011. Normal walking speed: A descriptive meta-analysis. *Physiotherapy* 97, 3 (2011), 182–189. DOI: https://doi.org/10.1016/j.physio.2010.12.004

[7] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. 2012. Crowdsourcing with smartphones. *IEEE Internet Comput.* 16, 5 (2012), 36–44. DOI: https://doi.org/10.1109/MIC.2012.70

[8] Constantinos Costa, Xiaoyu Ge, and Panos K. Chrysanthis. 2019. CAPRIO: Context-aware path recommendation exploiting indoor and outdoor information. In *Proceedings of the 20th IEEE International Conference on Mobile Data Management*. IEEE, 431–436. DOI: https://doi.org/10.1109/MDM.2019.000-7

[9] Constantinos Costa, Xiaoyu Ge, and Panos K. Chrysanthis. 2019. CAPRIO: Graph-based integration of indoor and outdoor data for path discovery. *Proc. VLDB Endow.* 12, 12 (2019), 1878–1881. DOI: https://doi.org/10.14778/3352063.3352089

[10] Constantinos Costa, Xiaoyu Ge, Evan McEllhenney, Evan Kebler, Panos K. Chrysanthis, and Demetrios Zeinalipour-Yazti. 2020. CAPRIO v2.0: A context-aware unified indoor-outdoor path recommendation system. In *Proceedings of the 21st IEEE International Conference on Mobile Data Management*. IEEE, 230–231. DOI: https://doi.org/10.1109/MDM48529.2020.00048

[11] Constantinos Costa, Brian T. Nixon, Sayantani Bhattacharjee, Benjamin Graybill, Demetrios Zeinalipour-Yazti, Walter Schneider, and Panos K. Chrysanthis. 2021. A context, location and preference-aware system for safe pedestrian mobility. In *Proceedings of the 22nd IEEE International Conference on Mobile Data Management*. IEEE, 217–224. DOI: https://doi.org/10.1109/MDM52706.2021.00042

[12] Buti Al Delail, Luis Weruaga, and M. Jamal Zemerly. 2012. CAViAR: Context aware visual indoor augmented reality for a university campus. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. IEEE Computer Society, 286–290. DOI: https://doi.org/10.1109/WI-IAT.2012.99

[13] Ugur Demiryurek and Cyrus Shahabi. 2017. *Predictive Path Planning*. Springer, 1630–1640. DOI: https://doi.org/10.1007/978-3-319-17885-1_1567

[14] Benjamin Ehret, Christian Henning, Maria R. Cervera, Alexander Meulemans, Johannes von Oswald, and Benjamin F. Grewe. 2021. Continual learning in recurrent neural networks. In *Proceedings of the 9th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=8xeBUgD8u9.

[15] Claudio Feliciani and Katsuhiro Nishinari. 2018. Measurement of congestion and intrinsic risk in pedestrian crowds. *Transport. Res. Part C: Emerg. Technol.* 91 (2018), 124–155. DOI: https://doi.org/10.1016/j.trc.2018.03.027

[16] Zijin Feng, Tiantian Liu, Huan Li, Hua Lu, Lidan Shou, and Jianliang Xu. 2020. Indoor top-k keyword-aware routing query. In *Proceedings of the 36th IEEE International Conference on Data Engineering*. IEEE, 1213–1224. DOI: https://doi.org/10.1109/ICDE48307.2020.00109

[17] Nimalika Fernando, David A. McMeekin, and Iain Murray. 2016. Poster: Context aware route determination model for mobile indoor navigation systems for vision impaired people. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion*. ACM, 24. DOI: https://doi.org/10.1145/2938559.2948799

[18] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. 2020. Quantifying SARS-CoV-2 transmission suggests epidemic control with digital contact tracing. *Science* 368, 6491 (2020). DOI: https://doi.org/10.1126/science.abb6936

[19] Matthew Garvey, Nilaksh Das, Jiaxing Su, Meghna Natraj, and Bhanu Verma. 2016. PASSAGE: a travel safety assistant with safe path recommendations for pedestrians. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, 84–87. DOI: https://doi.org/10.1145/2876456.2879470

[20] Qianyue Hao, Lin Chen, Fengli Xu, and Yong Li. 2020. Understanding the urban pandemic spreading of COVID-19 with real world mobility data. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 3485–3492. DOI: https://doi.org/10.1145/3394486.3412860

[21] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* 4, 2 (1968), 100–107. DOI: https://doi.org/10.1109/TSSC.1968.300136

[22] Jizhou Huang, Haifeng Wang, Miao Fan, An Zhuo, Yibo Sun, and Ying Li. 2020. Understanding the impact of the COVID-19 pandemic on transportation-related behaviors with human mobility data. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 3443–3450. DOI: https://doi.org/10.1145/3394486.3412856

[23] Søren Kejser Jensen, Jens Thomas Vejlby Nielsen, Hua Lu, and Muhammad Aamir Cheema. 2016. Outdoor-indoor space: Unified modeling and shortest path search. In *Proceedings of the 8th ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*. ACM, 35–42. DOI: https://doi.org/10.1145/3005422.3005427

[24] Richard L. Knoblauch, Martin T. Pietrucha, and Marsha Nitzburg. 1996. Field studies of pedestrian walking speed and start-up time. *Transport. Res. Rec.* 1538, 1 (1996), 27–38.

[25] Turgay Korkmaz and Marwan Krunz. 2001. Multi-constrained optimal path selection. In *Proceedings of the IEEE Conference on Computer Communications, 20th Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE Computer Society, 834–843. DOI: https://doi.org/10.1109/INFCOM.2001.916274

[26] Djibrilla Amadou Kountché, Benjamin Neveux, Nicolas Monmarché, Pierre Gaucher, and Mohamed Slimane. 2011. Indoor localization and guidance system for disabled people. In *Proceedings of the International Conference on the Network of the Future*. IEEE, 81–86. DOI: https://doi.org/10.1109/NOF.2011.6126688

[27] Fernando A. Kuipers, Turgay Korkmaz, Marwan Krunz, and Piet Van Mieghem. 2004. Performance evaluation of constraint-based path selection algorithms. *IEEE Netw.* 18, 5 (2004), 16–23. DOI: https://doi.org/10.1109/MNET.2004.1337731

[28] Fernando A. Kuipers, Piet Van Mieghem, Turgay Korkmaz, and Marwan Krunz. 2002. An overview of constraint-based path selection algorithms for QoS routing. *IEEE Commun. Mag.* 40, 12 (2002), 50–55. DOI: https://doi.org/10.1109/MCOM.2002.1106159

[29] Christos Laoudias, Artyom Nikitin, Panagiotis Karras, Moustafa Youssef, and Demetrios Zeinalipour-Yazti. 2021. Indoor quality-of-position visual assessment using crowdsourced fingerprint maps. *ACM Trans. Spatial Algor. Syst.* 7, 2 (2021), 10:1–10:32. DOI: https://doi.org/10.1145/3433026

[30] B. Li, J. P. Muñoz, X. Rong, Q. Chen, J. Xiao, Y. Tian, A. Arditi, and M. Yousuf. 2019. Vision-based mobile indoor assistive navigation aid for blind people. *IEEE Trans. Mob. Comput.* 18, 3 (Mar. 2019), 702–714. DOI: https://doi.org/10.1109/TMC.2018.2842751

[31] Mo Li, Pengfei Zhou, Yuanqing Zheng, Zhenjiang Li, and Guobin Shen. 2015. IODetector: a generic service for Indoor/Outdoor detection. *ACM Trans. Sens. Netw.* 11, 2 (2015), 28:1–28:29. DOI: https://doi.org/10.1145/2659466

[32] Yaguang Li and Cyrus Shahabi. 2018. A brief overview of machine learning methods for short-term traffic forecasting and future directions. *ACM SIGSPATIAL Special* 10, 1 (2018), 3–9. DOI: https://doi.org/10.1145/3231541.3231544

[33] Tiantian Liu, Zijin Feng, Huan Li, Hua Lu, Muhammad Aamir Cheema, Hong Cheng, and Jianliang Xu. 2020. Shortest path queries for indoor venues with temporal variations. In *Proceedings of the 36th IEEE International Conference on Data Engineering*. IEEE, 2014–2017. DOI: https://doi.org/10.1109/ICDE48307.2020.00227

[34] Halgurd S. Maghdid, Ihsan Alshahib Lami, Kayhan Zrar Ghafoor, and Jaime Lloret Mauri. 2016. Seamless outdoors-indoors localization solutions on smartphones: Implementation and challenges. *ACM Comput. Surv.* 48, 4 (2016), 53:1–53:34. DOI: https://doi.org/10.1145/2871166

[35] Felix Mata and Christophe Claramunt. 2014. A social navigation guide using augmented reality. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 541–544. DOI: https://doi.org/10.1145/2666310.2666364

[36] Felix Mata and Christophe Claramunt. 2015. A mobile trusted path system based on social network data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 101:1–101:4. DOI: https://doi.org/10.1145/2820783.2820799

[37] Mohamed F. Mokbel, Sofiane Abbar, and Rade Stanojevic. 2020. Contact tracing: Beyond the apps. DOI: https://doi.org/10.1145/3431843.3431846

[38] Brian T. Nixon, Sayantani Bhattacharjee, Benjamin Graybill, Constantinos Costa, Sudhir Pathak, Walter Schneider, and Panos K. Chrysanthis. 2021. HealthDist: a context, location and preference-aware system for safe navigation. In *Proceedings of the 22nd IEEE International Conference on Mobile Data Management*. IEEE, 250–253. DOI: https://doi.org/10.1109/MDM52706.2021.00050

[39] Seula Park, Seongyong Kim, and Kiyun Yu. 2018. Designing of indoor linkable pedestrian network data model for the transportation vulnerable. In *Proceedings of the 2nd International Conference on Digital Signal Processing*. ACM, 57–60. DOI: https://doi.org/10.1145/3193025.3193048

[40] Shakiba Rahimiaghdam, Pinar Karagoz, and Alev Mutlu. 2016. Personalized time-aware outdoor activity recommendation system. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 1121–1126. DOI: https://doi.org/10.1145/2851613.2851814

[41] Van Romero, William D. Stone, and Julie Dyke Ford. 2020. COVID-19 indoor exposure levels: An analysis of foot traffic scenarios within an academic building. *Transport. Res. Interdisc. Perspect.* 7 (2020), 100185. DOI: https://doi.org/10.1016/j.trip.2020.100185

[42] Hyeong-Gyu Ryu, Taehoon Kim, and Ki-Joune Li. 2014. Indoor navigation map for visually impaired people. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA'14)*. Association for Computing Machinery, New York, NY, 32–35. DOI: https://doi.org/10.1145/2676528.2676533

[43] Dimitris Sacharidis and Panagiotis Bouros. 2013. Routing directions: Keeping it fast and simple. In *Proceedings of the 21st SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 164–173. DOI: https://doi.org/10.1145/2525314.2525362

[44] Chaluka Salgado, Muhammad Aamir Cheema, and David Taniar. 2018. An efficient approximation algorithm for multi-criteria indoor route planning queries. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 448–451. DOI: https://doi.org/10.1145/3274895.3274938

[45] Thomas Springer. 2011. Mapbiquitous—An approach for integrated indoor/outdoor location-based services. In *Proceedings of the 3rd International Conference on Mobile Computing, Applications, and Services (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 95)*, Joy Ying Zhang, Jarek Wilkiewicz, and Ani Nahapetian (Eds.). Springer, 80–99. DOI: https://doi.org/10.1007/978-3-642-32320-1_6

[46] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 3104–3112. Retrieved from https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[47] Jesse Szwedko, Callen Shaw, Alexander G. Connor, Alexandros Labrinidis, and Panos K. Chrysanthis. 2009. Demonstrating an evacuation algorithm with mobile devices using an e-scavenger hunt game. In *Proceedings of the 8th ACM International Workshop on Data Engineering for Wireless and Mobile Access*. ACM, 49–52. DOI: https://doi.org/10.1145/1594139.1594154

[48] Xiaoqiang Teng, Deke Guo, Yulan Guo, Xiaolei Zhou, Zeliu Ding, and Zhong Liu. 2017. IONavi: An indoor-outdoor navigation service via mobile crowdsensing. *ACM Trans. Sens. Netw.* 13, 2 (2017), 12:1–12:28. DOI: https://doi.org/10.1145/3043948

[49] Qianru Wang, Bin Guo, Yan Liu, Qi Han, Tong Xin, and Zhiwen Yu. 2018. CrowdNavi: Last-mile outdoor navigation for pedestrians using mobile crowdsensing. *Proc. ACM Hum. Comput. Interact.* 2, CSCW (2018), 179:1–179:23. DOI: https://doi.org/10.1145/3274448

[50] Hang Wu, Suining He, and S.-H. Gary Chan. 2017. Efficient sequence matching and path construction for geomagnetic indoor localization. In *Proceedings of the International Conference on Embedded Wireless Systems and Networks*. Junction Publishing, Canada/ACM, 156–167. Retrieved from http://dl.acm.org/citation.cfm?id=3108030.

[51] Li Xiong, Cyrus Shahabi, Yanan Da, Ritesh Ahuja, Vicki Hertzberg, Lance Waller, Xiaoqian Jiang, and Amy Franklin. 2020. REACT: Real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking. *ACM SIGSPATIAL Special* 12, 2 (2020), 3–14. DOI: https://doi.org/10.1145/3431843.3431845

[52] Demetrios Zeinalipour-Yazti and Christophe Claramunt. 2020. COVID-19 Mobile Contact Tracing Apps (MCTA): A digital vaccine or a privacy demolition? In *Proceedings of the 21st IEEE International Conference on Mobile Data Management*. IEEE, 1–4. DOI: https://doi.org/10.1109/MDM48529.2020.00020

[53] Demetrios Zeinalipour-Yazti, Christos Laoudias, Kyriakos Georgiou, and Georgios Chatzimilioudis. 2017. Internet-based indoor navigation services. *IEEE Internet Comput.* 21, 4 (2017), 54–63. DOI: https://doi.org/10.1109/MIC.2017.2911420

[54] Anqi Zhao, Guanfeng Liu, Bolong Zheng, Yan Zhao, and Kai Zheng. 2020. Temporal paths discovery with multiple constraints in attributed dynamic graphs. *World Wide Web* 23, 1 (2020), 313–336. DOI: https://doi.org/10.1007/s11280-019-00670-4

[55] Tian Zhou, Lixin Gao, and Daiheng Ni. 2014. Road traffic prediction by incorporating online information. In *Proceedings of the 23rd International World Wide Web Conference*. ACM, 1235–1240. DOI: https://doi.org/10.1145/2567948.2580072

Received December 2020; accepted September 2021