ASTRO-K: Finding Top-k Sufficiently Distinct Indoor-Outdoor Paths

Vasilis Ethan Sarris
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA, USA
vas82@pitt.edu

Constantinos Costa
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA, USA
costa.c@cs.pitt.edu

Panos K. Chrysanthis
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA, USA
panos@cs.pitt.edu

2022 23rd IEEE International Conference on Mobile Data Management (MDM) | 978-1-6654-5176-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/MDM55031.2022.00083

***Abstract*—CAPRIO is an indoor-outdoor pedestrian path recommendation system that optimizes for shortest distance. Its path-finding algorithm, *ASTRO*, takes into account a set of user-provided congestion constraints and as such can recommend paths that can reduce the risk of COVID-19 exposure. In this paper, we extend *ASTRO* to consider the changes on congestion when providing path recommendations for overlapping requests. Our new algorithm, called *ASTRO-K*, can provide K alternative paths that satisfy the congestion constraints of all the path requests within a short time-window. Our experimental evaluation is conducted using two real-world datasets and shows that *ASTRO-K* can reduce the total average congestion of the recommended paths up to 4.5X with the trade-off of up to 7% increased total path time.**

***Index Terms*—Top-k Paths, Constraint-based Path Finding, Indoor-Outdoor Graphs, Congestion, COVID-19**

I. Introduction

The COVID-19 pandemic has shown us that there is a clear need for pedestrian path finding systems which can recommend paths that reduce the exposure to viral infection diseases. The exposure to viral airborne diseases is higher in crowded and congested spaces, and hence avoiding them reduces the risk of contracting a virus.

In our previous work, we proposed a solution to this problem of "physical distancing" with *ASTRO* [1], a constraint-based path-finding algorithm which factors in the predicted congestion of a space when constructing a path. *ASTRO* was implemented as the core path finding algorithm in CAPRIO [2]–[4], our indoor-outdoor pedestrian path recommendation system that suggests the shortest distance path for a given departure and arrival time between two locations.

By offering only one path per request, *ASTRO* may inadvertently contribute towards congestion problems by funneling people into a single area.

In this paper, we introduce *ASTRO-K*, an extension of *ASTRO* which computes the top-k sufficiently distinct constraint-satisfying paths for overlapping requests. Overlapping requests have similar departure and arrival time between the same two locations as shown in Figure 1. This provides CAPRIO with more paths to choose from for overlapping requests submitted within a short time-window and thus reduces the probability of inadvertently contributing towards congestion. Our hypothesis is that, *by only recommending paths which are sufficiently distinct, we are able to promote physical distancing and decrease the recommendations' contribution to congestion.*

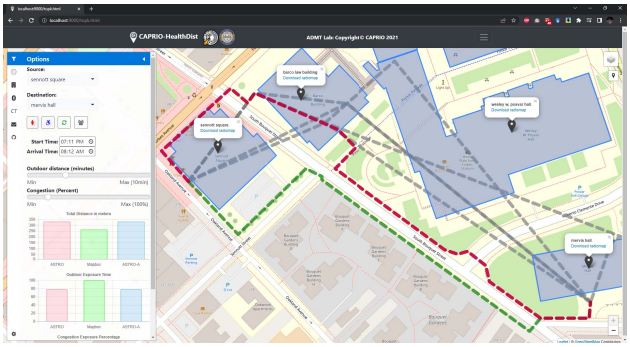


Fig. 1: CAPRIO with top-5 alternative paths produced by our *ASTRO-K*.

We evaluate *ASTRO-K* experimentally using two datasets modeling the campus of the University of Pittsburgh and the University of Cyprus. Our experiments show that *ASTRO-K* can reduce the average congestion by increasing K. Specifically, the average congestion of the recommended paths is reduced by 4.5X for the PITT dataset with $K = 6$ and 2.6X for the UCY dataset with $K = 3$. This reduction in the average congestion comes with the cost for the shortest distance. We observe an increase of total path time up to 7%. Clearly, these initial results are promising and support our hypothesis above.

Our main contributions are summarized as follows:

- We introduce *ASTRO-K*, an integration of the *ASTRO* and *ESX* [5] algorithms, which to our knowledge is the first algorithm for finding top-k sufficiently distinct constraint-satisfying paths.
- We conduct a preliminary experimental evaluation and show that *ASTRO-K* can reduce the total average congestion produced by the recommendations.

The rest of the paper is structured as follows. In the next section, we formalize the problem and provide the necessary background on *ASTRO* and *ESX* algorithms. In Section III, we present *ASTRO-K* and in Sections IV and V the experimental methodology and its evaluation respectively. In Section VI, we discuss related work.

2375-0324/22/$31.00 ©2022 IEEE
DOI 10.1109/MDM55031.2022.00083

II. Problem Formulation & Background

In this section, we define all the relevant information necessary to understanding the *ASTRO-K* algorithm. This includes a formulation of the problem (Section II-A), an explanation of the underlying graph (Section II-B), definitions of the constraints used by *ASTRO* and *ASTRO-K* (Section II-C), and an overview of the *ASTRO* path finding algorithm (Section II-D).

A. Problem Formulation

Given an Indoor-Outdoor graph, a source Outdoor vertex, a terminal Outdoor vertex, a set of constraints, find the top-k sufficiently distinct paths from the source outdoor vertex to the terminal outdoor vertex which satisfy the given constraints.

B. Indoor-Outdoor Graphs

In order to understand the *ASTRO-K* algorithm, it is first helpful to understand the underlying graph upon which the algorithm is built. An Indoor-Outdoor graph G is constructed using three main components: Indoor vertices, Indoor graphs, Outdoor vertices. Indoor vertices are the bottom layer of an Indoor-Outdoor graph and represent the doors allowing people to enter and exit buildings. Indoor graphs are bidirectional weighted graphs which are comprised of Indoor vertices and edges represent the path between any two doors. Outdoor vertices are comprised of Indoor graphs and represent buildings. Indoor-Outdoor graphs are bidirectional weighted graphs comprised of Outdoor vertices and edges represent the paths between buildings. Thanks to the nature of an Indoor-Outdoor graph, *ASTRO* is able to seamlessly find paths inside and outside of buildings.

C. Constraints

Definition 1: Outdoor Exposure (E) – the maximum amount of time allowed to traverse any given edge between two outdoor vertices.

Definition 2: Time Limit (T) – the maximum amount of time allowed to traverse a path.

Definition 3: Congestion (C) – the maximum amount of congestion which can be encountered while traversing between two indoor vertices.

Definition 4: Path Similarity (θ) – the maximum percentage of total edge distance shared between any two paths in the result set.

D. ASTRO

Before describing *ASTRO-K*, let us first review the original algorithm, *ASTRO*. *ASTRO* is a constraint-based variant of the A^* algorithm. It is an optimal A^* variant, which means that *ASTRO* is guaranteed to have optimal edge selection when traversing the graph [1].

Given an Indoor-Outdoor graph, *ASTRO* behaves like standard A^*, traversing the graph over the Outdoor vertices but then at each step also expands the current Outdoor vertex's Indoor graph to find the best ordered pair of Indoor vertices to use as the entry and exit for the current Outdoor vertex. Unlike standard A^* when an edge is expanded it may be pruned if it no longer meets the constraints $\Pi = (E, T, C)$. Pruning edges in this manner allow us to avoid constructing the complete Indoor-Outdoor graph which would be prohibitively expensive.

Algorithm 1 Modified ASTRO used in Algorithm 2

Input: s: source, t: terminal, O: Outdoor Vertices, E_R: Edges Removed, Π: Constraints;
Output: p: Best Path;

```
init Priority Queue OPEN, Set CLOSED
init Outdoor Vertex start with s
OPEN.push(start)
while OPEN not empty do
  curr ← OPEN.pop()
  if curr = NULL then
    return ReconstructPath(curr)
  if curr ∉ CLOSED then
    CLOSED ← CLOSED ∪ {curr}
    for o_i ∈ {O − CLOSED} do
      IndoorGraph ← o_i[IndoorGraph] − E_R
      for in_i ∈ IndoorGraph do
        s1⃗ ← Status(curr[out], in_i, curr[g])
        for out_i ∈ {IndoorGraph − {in}} do
          now = curr[g] + s1⃗[total]
          s2⃗ ← Status(in_i, out_i, now)
          s⃗ ← s1⃗ + s2⃗
          g ← curr[g] + s⃗[total]
          if g < o_i[g] and Check(Π, s⃗) then
            init Outdoor Vertex toAdd
            OPEN.push(toAdd)
```

It is key to note that the unit of measure for both the exact cost function $g()$ and estimated cost heuristic $h()$ is *time* rather than distance. The exact cost $g()$ for an Outdoor vertex is the sum of the time it takes to traverse to the current vertex, and the heuristic cost $h()$ is an estimate of the amount of time it will take to reach the terminal vertex from the current vertex. This change in unit of measure is accomplished by simply multiply the distance by the average walking speed of a person ($1.4m/s$) [6], [7]. This change although simple is critical to the algorithm as it allows *ASTRO* to dynamically compute predicted congestion based on the estimated time of arrival. Additionally, when indoor graphs are constructed, the indoor edge weights are modified to account for the delay due to congestion which is modeled as a function of the percentage of the original indoor travel time. For example, if the original indoor time $i = 10s$ and there is 50% congestion, then the added congestion delay is 5s.

E. ESX

In contexts where distance is not the only factor which need to be considered, such as within *ASTRO-K*, simply finding the top-k shortest paths may not be the best option. Chondrogiannis et al. [5] defined the K Shortest Paths with Limited Overlap problem, showed it was weakly NP-hard, and proposed a number of solutions including the *ESX* algorithm.

ESX is a performance-oriented heuristic algorithm that given a graph G, a number of paths to find K and maximum similarity ratio θ will compute the K shortest paths that are at most θ similar. The algorithm essentially finds the best path, begins to iteratively remove the costliest edge from the previously found path and searches for the best path using the updated graph until either: K paths are found or every possible edge within the path it previously found was attempted to be removed and thus can't continue. The similarity of paths p_1 and p_2 can be defined as $similarity = (E_{p_1} \cap E_{p_2})/E_{p_1}$ where E_{p_i} is the edge set of path i.

III. *ASTRO-K*

The major contribution of this paper is an extension of the *ASTRO* path finding algorithm, dubbed *ASTRO-K*, which enables the efficient computation of the top-k sufficiently distinct paths for a given Indoor-Outdoor graph. This was achieved via the integration of a modified *ASTRO* and a modified version of the *ESX* algorithm.

While the idea behind *ESX* is perfect for our need to offer multiple different enough paths to disperse congestion, we are unable to modify the underlying Indoor-Outdoor graph. Because of this constraint, both the *ASTRO* and *ESX* algorithms were modified to suit these needs. *ESX* was modified to leave the Indoor-Outdoor graph unchanged and instead achieve the same result using a Removed Edge set, which is passed through to *ASTRO-K*'s version of *ASTRO*. *ASTRO-K*'s version of *ASTRO* was then modified to use this Removed Edge set.

The aspect of *ASTRO-K* employing a version of *ESX* can be found in Algorithm 2. To begin, we initialize the result set P with the best path given the constraints found using *ASTRO* (Lines 1-2) and then proceed to add each edge of the path found into the priority queue PQ (Lines 3-4). PQ is a min priority queue based on the total time cost of the edge. From here on (Lines 5-20) the algorithm loops until K paths have been found or there are no more edges in the priority queue PQ to remove from the graph. The loop starts by initializing the current path p with the last path added to the result set P (Line 6). Lines 7-15 loop until the PQ has run out of edges to remove or a path that is sufficiently dissimilar is found. This nested loop first initializes the current edge by popping off the PQ (Line 8), makes sure it's not within the Do Not Remove Edge set E_{DNR} (Lines 12-13), adds the edge to the Removed Edge set E_R (Line 11) and then calls *ASTRO* to find the best path given the removed edges and constraints (Line 12). If *ASTRO* does not return a valid path, we remove the edge from the Removed Edge set E_R and add it to the Do Not Remove Edge set E_{DNR} (Lines 13-15). Once the nested while loop is broken we must check if a suitable path was found (Line 16), if there was, add it to the result set and reinitialize the priority queue with the edges of the path we just found (Lines 17-20). If this was not the case and the priority queue was emptied, the outer loop will break and an incomplete set of result paths will be returned. Note this does not mean that no more sufficiently distinct constraint-satisfying paths exist, this solution is adapted from a performance-oriented heuristic and is thus not complete.

Algorithm 2 *ASTRO-K*

Input: s: source, t: terminal, O: Outdoor Vertices, Π: Constraints;
Output: P: Best Paths

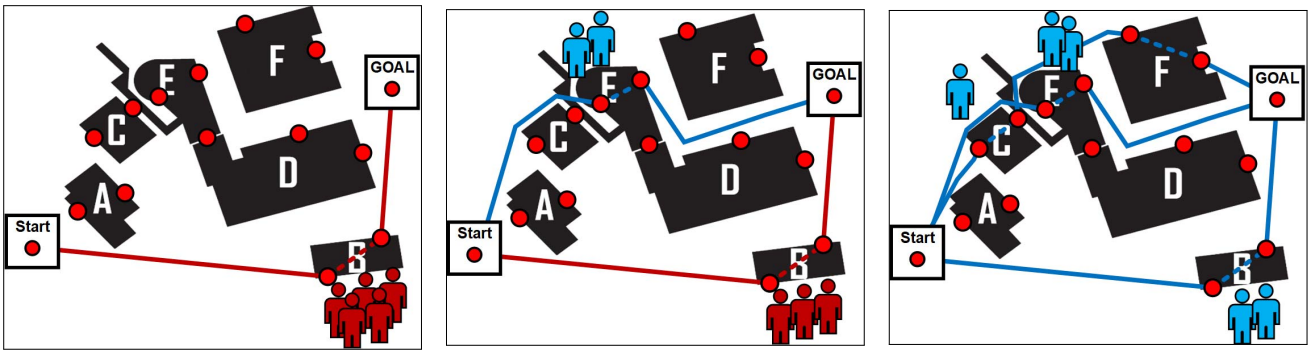
1: init List P, Priority Queue PQ, Set E_{DNR}, Set E_R
2: $P.append(ASTRO(s, t, O, E_R, \Pi))$
3: **for** $e \in P.tail()$ **do**
4: $\quad PQ.push(e)$
5: **while** $|P| < k$ **and** PQ not empty **do**
6: $\quad p \leftarrow P.tail()$
7: $\quad$ **while** PQ not empty **and** $Sim(p, P) > \theta$ **do**
8: $\quad\quad e \leftarrow PQ.pop()$
9: $\quad\quad$ **if** $e \in E_{DNR}$ **then**
10: $\quad\quad\quad$ **continue**
11: $\quad\quad E_R \leftarrow E_R \cup \{e\}$
12: $\quad\quad p \leftarrow ASTRO(s, t, O, E_R, \Pi)$
13: $\quad\quad$ **if** $p = NULL$ **then**
14: $\quad\quad\quad E_R \leftarrow E_R - \{e\}$
15: $\quad\quad\quad E_{DNR} \leftarrow E_{DNR} \cup \{e\}$
16: $\quad$ **if** $Sim(p, P) < \theta$ **then**
17: $\quad\quad P.append(p)$
18: $\quad\quad PQ.clear()$
19: $\quad\quad$ **for** $e \in p$ **do**
20: $\quad\quad\quad PQ.push(e)$
21: **return** P

As described earlier, the implementation of *ASTRO* described in Algorithm 1 which is used by *ASTRO-K* is varied from the original. These variations can be seen via the input parameters and the Indoor graph expansion. First, the parameters have been expanded to include the Removed Edge set E_R, this allows for their use later on in the algorithm. Second, when expanding an Outdoor vertices' Indoor graph and initializing out local copy (Line 11), the edges from the Removed Edge set E_R are subtracted from the local copy rather than the global Indoor-Outdoor graph. This allows us to keep the global state identical while path finding with edges removed.

Example: To showcase *ASTRO-K* in action, consider the set of buildings ($\{A, B, C, D, E, F\}$) in Figure 2 where five people want to find a path from the start to the goal, with congestion tolerance level 2 (persons). Also, assume that initially the congestion level in all buildings is 0. In the graph, both start and goal are outdoor vertices with only one indoor vertex. The buildings represent the outdoor vertices, and the red dots are indoor vertices, which represent the doors.

Figure 2(a) shows that the *ASTRO-K* with $K = 1$ recommends the same path for all five people, which has as a result that the outdoor vertex B will become congested at level 5 since all of them are passing through that building in a short period of time. This clearly violates the congestion constraint.

In the case of $K = 2$, *ASTRO-K* recommends two distinct paths, which reduce the congestion of outdoor vertex B, yet



(a) *ASTRO-K* with $K = 1$ (b) *ASTRO-K* with $K = 2$ (c) *ASTRO-K* with $K = 3$

Fig. 2: *ASTRO-K* recommends 3 paths that can reduce the congestion based on the parameter $K = 3$ to 5 people with congestion tolerance level 2. For example, if 5 people request a path, *ASTRO-K* will disperse the people into K paths.

violate the congestion constraint even though only three of five persons are passing through that building, and the other two people are passing through building E as shown in Figure 2(b).

Finally, *ASTRO-K* with $K = 3$ recommends three paths resulting in a distribution of congestion where two persons are passing through building B, two through E, and one through C and F, hence meeting all five persons' congestion constraints.

IV. Experimental Methodology and Evaluation

This section provides details regarding the algorithm, testbed, datasets, and metrics used for the evaluation of our *ASTRO-K* algorithm.

Algorithm: ASTRO-K finds the top-k sufficiently distinct constraint-satisfying paths so that we can spread the congestion generated by best path recommendations. We evaluate *ASTRO-K*'s performance in terms of average congestion and number of people needed to violate the constraints while varying the parameter K. We chose the origin and destination points to be the points with the maximum euclidean distance in each topology created by the following datasets.

Datasets:

- **PITT**: This is a realistic dataset that was created using the University of Pittsburgh campus and 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**: This is a realistic dataset that was created using the University of Cyprus campus and 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.

Testbed: Our evaluation is carried out on a dedicated machine with Manjaro Linux. The server is featuring 16 GB of RAM with 8 Cores (@ 1.80GHz), a 500 GB SSD.

Congestion Generation: We used camera analysis [8] on a 2-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 EPICGen generator [9] with default experimental parameters for simulating both pass-through and scheduled traffic.

Metrics: Due to the limited pre-existing work on this exact problem defined in Section II-A, we chose the following two metrics to evaluate the performance and showcase the importance of *ASTRO-K*.

- **Average Congestion (*C*):** measures the average percentage of congestion within a $3m^2$ cell of a building during a 5-minute window. Percentage of congestion is determined with respect to the maximum amount of people allowed within a cell.
- **Number of People (*P*):** measure the amount of people introduced within a 5-minute window.

Methodology: All of our experiments look the effect of K on estimating C if 50 people were introduced into the Indoor-Outdoor graphs modeling the *PITT* and *UCY* datasets. Particularity, the people are assigned to K paths using a priority queue based on the minimum average congestion of each path and adjusted after every assignment.

Let D be the average door-to-door distance for a dataset, w be the constant walking speed of $1.4m/s$, ϕ be the $3m^2$ cell size used in our congestion generator and S be the 5-minute window for which congestion is predicted. For all experiments, we model the congestion increment of a single person to a path as $inc = \frac{D}{w*\phi*S}$, which can be understood semantically as the average congestion a person will add to each grid cell within a building with respect to a 5-minute window of time.

V. Experimental Results

We conducted three experiments to assess the effectiveness of *ASTRO-K* in reducing the congestion and measure the impact on the shortest distance/time when recommending its returned distinct paths.

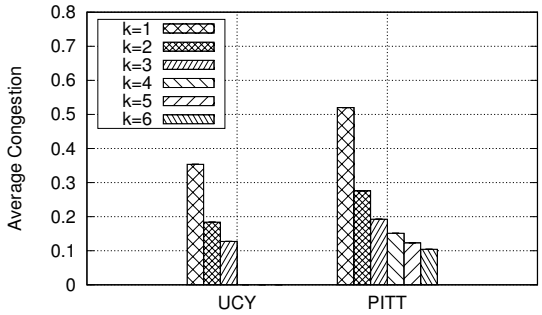


Fig. 3: The average congestion of all the recommended paths when K paths are found.

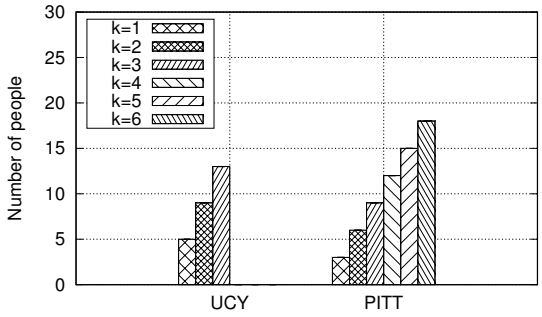


Fig. 4: The number of people which were added to a network before a path breaches the congestion constraint when K paths are found.

A. Experiment 1: Average congestion per path

In the first experiment we study the average congestion per path in respect with the number of recommended paths (K). In Figure 3, we can clearly see that as the number of paths found increases, the average C value for the K paths dramatically reduces. Specifically, the average congestion of the recommended paths is reduced by 4.5X for the PITT dataset with $K = 6$ and 2.6X for the UCY dataset with $K = 3$.

Summary: ASTRO-K is able to effectively disperse congestion introduced by recommended paths.

B. Experiment 2: Number of people

In the second experiment, we look at the maximum number of people *P* going from the same source vertex to destination vertex that we are able to introduce into the Indoor-Outdoor graphs for *PITT* and *UCY* datasets before the average congestion reaches a constraint of 15% in respect with the number of recommended paths (K). In Figure 4, we observe that as K increases there is a clear increase in the amount of people, which are able to be introduced into the paths while not breaching the congestion constraint. Particularly, the number of people which can follow the recommended paths is increased by 5.9X for the PITT dataset with $K = 6$ and 2.8X for the UCY dataset with $K = 3$.

Summary: ASTRO-K is able to drastically increase the amount of people we are able to recommend paths to without exceeding the constraints.

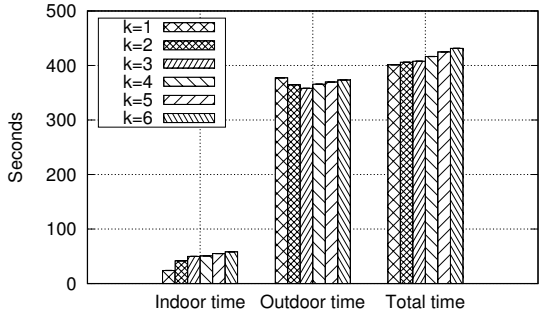


Fig. 5: The average time of all recommended paths when K paths are found.

TABLE I: Average indoor, outdoor, and total time for 50 people varying K.

Time	k=1	k=2	k=3	k=4	k=5	k=6
Indoor	23.94	41.62	49.88	50.54	55.09	58.12
Outdoor	377.215	364.52	358.12	365.88	369.60	373.25
Total	401.16	406.14	408.01	416.43	424.70	431.38

C. Experiment 3: Average time for all recommended paths

In this third experiment, we examine the average indoor, outdoor, and total time of all the paths for 50 people in respect with the number of recommended paths (K) using PITT dataset to determine the trade-off of increasing K. Figure 5 and Table I show that as the number of paths found increases, the average total time for the K paths increases as well. Specifically, the average total time of the recommended paths is increased by 7% for $K = 6$. Furthermore, the average indoor time increases by 2.4X and the average outdoor time decrease by 2% for $K = 6$.

Summary: ASTRO-K is able to effectively disperse congestion with the trade-off of slightly increased total path time.

VI. Related Work

In this section, we discuss the most recent strongly related work of ASTRO-K. A taxonomy of this related work is shown in Table II.

Top-k Sufficiently Distinct path finding: As described in Section II-E, ASTRO-K adopted one of the approaches proposed by Chrondrogiannias et al. to tackle this problem. Liu et al.'s [12] also proves a formalization of the problem to be NP-Hard and proposes a similar solution to the ESX algorithm while performing an additional diversity lower bound calculation at each step.

Constraint-based Path Finding: A formalization of constraint-based path finding known as the Multi-Constraint Path finding (MSP) problem is explored in Feng & Korkmaz [10] and Hu et al. [11]. Feng & Korkmaz provide a comprehensive overview of the previous work on the problem as well as propose two multi-constraint path finding algorithms

TABLE II: Taxonomy of Related Work

Paper	Top-k	Constraints	Congestion
ASTRO [1]	No	Yes	Predictive
Feng & Korkmaz [10]	Partially	Yes	No
Hu et al. [11]	No	Yes	No
Liu et al. [12]	Yes	No	No
Saleem et al. [13]	No	No	Real-Time
Walied et al. [14]	No	No	Real-Time
Lin et al. [15]	No	No	Real-Time
Shreyas et al. [16]	No	No	Real-Time
Liu et al. [17]	No	No	Predictive
ASTRO-K	Yes	Yes	Predictive

which are able to find multiple paths. Hu et al. proposes a more efficient MSP algorithm but is constricted to a single-path. While MSP may be a similar problem it is not applicable to ASTRO because both approaches would necessitate full graph expansion which we explicitly want to avoid do to how computationally expensive it is to fully construct the graph.

Congestion-Aware Indoor path finding: Congestion-aware indoor path finding is a topic which is explored in many contexts and thus most works, while potentially similar in concept, are trying to solve fundamentally different problems. Indoor-localization [13], [14] and IoT [15], [16] approaches are good examples of this. These are both efficient indoor path finding approaches but are not applicable in the context of predictive path finding [18] as performed by ASTRO and ASTRO-K due to their real-time data updating the environment.

Liu et al. [17] proposes an algorithm for the same kind of door-to-door congestion-aware path finding. While ASTRO/ASTRO-K use a grid-based building model to predict congestion within a grid cell, Liu et al. predict congestion by modeling buildings in terms of the semantics of a space and then simulating the movement of people via queues hyper-parameterized using historical data. However, since the grid cells designate a much smaller space than Liu et al.'s semantic modeling approach, ASTRO/ASTRO-K is able to provide paths considering congestion at a much finer granularity.

VII. Conclusions & Future Work

In the recent years, and more precisely since the outbreak of the COVID-19 pandemic, the impact of crowded and congested spaces on the spread of viral airborne diseases attracted greater attention, which go beyond accessibility. In this paper, we present a novel top-k sufficiently distinct extension of *ASTRO*, dubbed *ASTRO-K*, which enables us to recommend paths to more people without inadvertently congesting an area. *ASTRO-K* is stateless, i.e., does not maintain a history of path recommendations. It integrates a slight modification of the *ESX* algorithm along with a modified version of our previous work in order to accomplish this. Our experimental results show that *ASTRO-K* is able to recommend paths in a manner that reduces the average congestion of the paths.

Our proposed solution does not include balancing or scheduling of route recommendations. This is the next step of our work. We plan to use *ASTRO-K* as a core component in the development of path scheduling algorithms that can recommend paths in a setting where multiple path discovery requests need to be considered together. Finally, we also plan to make the code open source as part of the CAPRIO[1] project.

VIII. Acknowledgments

This paper was part of the capstone of the first author. This work was partially funded by NIH award U01HL137159 and by the Pittsburgh Foundation. 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] C. Anastasiou, C. Costa, P. K. Chrysanthis, C. Shahabi, and D. Zeinalipour-Yazti, "ASTRO: reducing COVID-19 exposure through contact prediction and avoidance," *ACM Trans. Spatial Algorithms Syst.*, vol. 8, no. 2, pp. 1–31, 2022.

[2] C. Costa, X. Ge, and P. K. Chrysanthis, "CAPRIO: context-aware path recommendation exploiting indoor and outdoor information," in *IEEE Intl. Conf. on Mobile Data Management*, 2019, pp. 431–436.

[3] ——, "CAPRIO: graph-based integration of indoor and outdoor data for path discovery," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1878–1881, 2019.

[4] C. Costa, X. Ge, E. McEllhenney, E. Kebler, P. K. Chrysanthis, and D. Zeinalipour-Yazti, "Capriov2.0: A context-aware unified indoor-outdoor path recommendation system," in *IEEE Intl. Conf. on Mobile Data Management*, 2020, pp. 230–231.

[5] T. Chondrogiannis, P. Bouros, J. Gamper, U. Leser, and D. B. Blumenthal, "Finding k-shortest paths with limited overlap," *VLDB J.*, vol. 29, no. 5, pp. 1023–1047, 2020.

[6] R. Bohannon, A. W. Andrews, "Normal walking speed: a descriptive meta-analysis," *Physiotherapy*, vol. 97, no. 3, pp. 182–189, 2011.

[7] R. L. Knoblauch, M. T. Pietrucha, and M. Nitzburg, "Field studies of pedestrian walking speed and start-up time," *Transportation Research Record*, vol. 1538, no. 1, pp. 27–38, 1996.

[8] C. Feliciani and K. Nishinari, "Measurement of congestion and intrinsic risk in pedestrian crowds," *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 124–155, 2018.

[9] C. Anastasiou, C. Costa, P. K. Chrysanthis, and C. Shahabi, "Epicgen: An experimental platform for indoor congestion generation and forecasting," *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 2803–2806, 2021.

[10] Gang Feng, Turgay Korkmaz, "Finding multi-constrained multiple shortest paths," *IEEE Trans. Computers*, vol. 64, no. 9, pp. 2559–2572, 2015.

[11] X. Hu, K. Wang, J. Wang, K. Wang, Y. Hu, and S. Wang, "Multi-constrained routing optimization algorithm based on DAG," in *Conf. of the IEEE Industrial Electronics Society*, 2018, pp. 5906–5910.

[12] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k shortest paths with diversity," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 488–502, 2018.

[13] A. Saleem, K. A. Jabri, A. A. Maashri, W. A. Maawali, and M. Mesbah, "Obstacle-avoidance algorithm using deep learning based on rgbd images and robot orientation," in *Intl. Conf. on Electrical and Electronics Engineering*, 2020, pp. 268–272.

[14] A. M. Walied, A. Onsy, S. A. Maged, and S. Hammad, "Path planning in a dynamic indoor environment for mobile robots using q-learning technique," in *Intl. Mobile, Intelligent, and Ubiquitous Computing Conf.*, 2021, pp. 373–380.

[15] C. Lin, G. Han, J. Du, T. Xu, L. Shu, and Z. Lv, "Spatiotemporal congestion-aware path planning toward intelligent transportation systems in software-defined smart city iot," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8012–8024, 2020.

[16] S. J, H. Singh, J. Bhutani, S. Pandit, S. N. N, and D. Kumar S M, "Congestion aware algorithm using fuzzy logic to find an optimal routing path for iot networks," in *Intl. Conf. on Computational Intelligence and Knowledge Economy*, 2019, pp. 141–145.

[17] T. Liu, H. Li, H. Lu, M. A. Cheema, and L. Shou, "Towards crowd-aware indoor path planning," *Proc. VLDB Endow.*, vol. 14, no. 8, pp. 1365–1377, 2021.

[18] U. Demiryurek and C. Shahabi, "Predictive path planning," in *Encyclopedia of GIS*. Springer, 2017, pp. 1630–1640.

[1] CAPRIO: https://db.cs.pitt.edu/caprio/