Mobile-based Urban Bike Route Planner using Urban Regulation-constrained Delaunay Graph

Juan Gutierrez-Urrego
Jorge Correa
Placid Ferreira
jcgutierru@eafit.edu.co
jorge.correa@cohesivemanufacturing.com
ceo@cohesivemanufacturing.com
Cohesive Manufacturing
Champaign, Illinois, USA

Saul Rivera-Betancur
saul.rivera@metropol.gov.co
Area Metropolitana
Valle de Aburra, Colombia

Oscar Ruiz-Salguero
oruiz@eafit.edu.co
U. EAFIT
Colombia

ABSTRACT

In the domain of bike route planning for urban environments, the solutions provided by large corporations (e.g. Google Maps, Waze-Google) are not tailored for this particular vehicle or do not reflect path cost structures that human interactions and agglomerations produce. Bikepath expenses different from the usual Euclidean or City-Block distance functions but relevant in a city relate to safety (in terms of accidents or criminality), slopes, path roughness, time-dependent (i.e. rush hour) costs, etc. To partially overcome these disadvantages, this manuscript presents the implementation of a bike route planning algorithm in a urban environment, which efficiently solves the problem of presenting the biker with a low cost route. At the same time, our application allows flexibility in the degree of usage of dedicated bike routes built by the city. This flexibility obeys to city regulations, which may prescribe more or less priority in the usage of dedicated bikepaths. Our algorithm integrates bike dispensers, bike routes, variety of costs (additional to travel length) and finds the suggested routes in a constrained Delaunay graph. The execution of the algorithm is enhanced by using the fact that large part of the travel might be pre-computed if the biker must pick up and return the city-provided bikes in specific dispenser points. Future work is needed in (a) adding more flexible heuristics as the city may decide to prioritize diverse environmental, economic, or transportation goals, (b) transcending canonical metrics, e.g. by considering non-symmetrical costs ($d(p,q) \neq d(q,p)$).

CCS CONCEPTS

• **Theory of computation** → *Design and analysis of algorithms*; • **Software and its engineering** → *Software creation and management*; • **Information systems** → **Web applications**; *Data management systems.*

KEYWORDS

Delaunay Triangulation, Graph Constraints, Urban Route Planning, Sustainable Transportation, Small City

Web3D '23, October 09–11, 2023, San Sebastian, Spain

ACM ISBN 979-8-4007-0324-9/23/10…$15.00
https://doi.org/10.1145/3611314.3615921

ACM Reference Format:
Juan Gutierrez-Urrego, Jorge Correa, Placid Ferreira, Saul Rivera-Betancur, and Oscar Ruiz-Salguero. 2023. Mobile-based Urban Bike Route Planner using Urban Regulation-constrained Delaunay Graph. In *The 28th International ACM Conference on 3D Web Technology (Web3D '23), October 09–11, 2023, San Sebastian, Spain.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3611314.3615921

GLOSSARY

Ω	Connected urban territory ($\Omega \subset \mathbb{R}^2$) with 2-manifold topology. $p \in \Omega$ has the form $p =$ (`longitude`, `latitude`).
$\partial\Omega$	Border of Ω whose components Γ_i have 1-manifold topology. $\partial\Omega = \{\Gamma_0, \Gamma_1, ..., \Gamma_n\}$ with Γ_0 being the outermost border of Ω and Γ_i ($i \geq 1$) being the inner borders (e.g. pond shores).
$C_i(u)$	Piecewise Linear continuous curves, representing bikepaths, with its sampled points in Ω. Bikepaths C_i may be self-intersecting if it contains overpasses above itself. Bikepaths C_i and C_i may share vertices.
G_C	Graph $G_C = (V_C, E_C) = \bigcup_i C_i(u)$. The non-directed graph G_C contains all the vertices and edges of all bikepaths $C_i(u)$. Edge $(v_i, v_j) = (v_j, v_i) \in E_C$ if v_i and v_j are consecutive point samples of a bikepath curve $C_i(u)$.
D	$D = \{v_{d1}, v_{d2}, ...\}$ is the set of bike dispensers. Computing of edge connections from D to G_C is required as bikepaths do not necessarily reach bike dispensers.
G_B	Graph $G_B = (V_B, E_B)$ with V_B being vertices either (a) in V_C or (b) bike dispensers. E_B are non-directed Edges on V_B whose flexible cost function is discussed. $G_B = G_C \cup D$.
G	Fully connected urban bike route Graph $G = (V_B, E)$ with edges E being either (a) Edges of C_i bikepaths, (b) bridges from bike dispensers D to bikepaths C_i, or, (c) Delaunay-related Edges which integrate all metropolitan bikepaths. This is the graph on which user-requested bike rides are computed.
H	Convex Hull of the set of sites V_B (bike dispensers D and bikepath vertices set V_C).
n_C	number of vertices of the bikepathways $\{C_1, C_2, ..., C_n\}.s$
n_D	number of bike dispensers ($n_D = \|D\|$).
n_B	number of vertices of the bikepathways plus bike dispenser graphs. $n_B = n_C + n_D - \|D \cap V_C\|$.
P	$(n_D \times n_D)$ matrix of optimal path vertex sequences.

PL	Piecewise Linear. In this manuscript, the bikepaths C_i are PL (straight - segment) curves in $\mathbf{R}^2$.
R	Rides matrix. Strictly triangular matrix that contains the computed bike rides, Google Maps® walking paths, and their respective distance cost.
$DT(V)$	Delaunay Triangulation of a set of vertices V, in this case scattered in $\mathbf{R}^2$.
2-manifold	a surface point set locally isomorphic to a flat disk (i.e. without self-intersections).
1-manifold	a curve point set locally isomorphic to a straight wire (i.e. without self-intersections).

1 INTRODUCTION

Bike ride optimization is obviously central in sustainable urban transportation. Current bike route planners are based on graph structures equipped with Euclidean or City-block metrics. Bike rides demand lesser specifications from the physical path than car routes do. Therefore, bike trajectory graphs may be considerably larger than their car counterparts. Thus, the reduction of a search space is an important requirement for any bike route planner.

A graph $G = (V, E)$ is a set of vertices $V = \{v_1, v_2, ..\}$ communicated by a set of edges $E = \{(v_i, v_j), (v_m, v_k), ...\}$ with a cost $f : E \rightarrow \mathbb{R}$. The *topology* of G is concentrated in its connectivity E. In spatially embedded graphs, the *geometry* of G may refer to (a) the coordinates of the site where a vertex v_i resides, or (b) in a rarely used manner, the costs of the edges $e_k = (v_k, v_w) \in E$, or other property such as size, cost, attenuation, pattern, etc., that grades the edges $e \in E$.

An oblique manner to modify the topology of G by tuning its geometry is to set an expense to transit an edge as high as needed to make that edge e unusable ($f(e) \rightarrow \infty$). In this manner, the edge becomes un-affordable and thus nearly non-existent. However, we prefer to directly create the topology of the bike ride graphs by applying city regulation heuristics.

In this manuscript, the application of heuristics has diverse purposes (one of them being graph topology definition). We refer in the text to the *heuristics* that follow.

1.1 Urban Heuristics

Refer to the bias introduced into our geometric algorithms by urban regulations. In this manuscript, for example, we consider a regulation that prioritizes the usage of bike-dedicated paths. This heuristic impacts on the solution of conflicts between bike-dedicated paths and Delaunay trajectories (edges). There could be other factors such as path quality, path risks, rush hour congestion, air pollution, etc. However, we do not consider all these factors at this time.

1.2 Geometric Heuristics

Refer to the usual programming decisions that make a solution affordable and useful in most of the cases, but which has no theoretical guarantee of correction or optimality.

Consistent with these considerations, we report here the Mobile implementation of the EnCicla® path planner for urban bike rides, with (a) flexible path cost definition, (b) flexible prioritization of (bike-dedicated vs. car-shared) bikepaths, (c) low computing resource demand.

EnCicla®App has been deployed in the metropolitan area called *Valle de Aburra* (Colombia). This metropolitan area encompasses the following cities: Medellin, Envigado, Sabaneta, Itagui, Estrella, Caldas, Bello, Girardota, Copacabana and Barbosa, with a population approaching 4 million people, including 130000 registered customers of bike dispensers. The resulting non-directed graph represents 190 km of bikepats, 110 bike dispenser sites, 7000 bike system vertices and 9000 edges.

In this manuscript, Section 2 reviews the existing literature and draws conclusions for the implementation. Section 3 describes the methodology applied. Section 4 discusses some results of the software operation. Section 5 concludes the manuscript and mentions possible lines of future work.

2 LITERATURE REVIEW

2.1 Commercial Systems

2.1.1 [Ferster et al. 2020]. The article explores the use of OpenStreetMap (OSM) data for inventorying bicycling infrastructure in Canadian cities. While OSM data is valuable, there may be discrepancies compared to municipal open data. On-street bicycle lanes show higher concordance, while cycle tracks and local street bikepaths have lower concordance due to their newness or rarity in some cities.

2.1.2 [Ariyanto et al. 2022]. The GoWes website-based cycling navigation in Batu city offers personalized route choices and accurate suggestions. Yet, there are areas for enhancement. These encompass lacking a mobile app, no user authentication, and reliance on data accuracy. Moreover, real-time traffic updates and multimode integration are absent. Addressing these could enrich the user experience and offer a more holistic cycling navigation.

2.2 Edge Types and Weighted Routing

2.2.1 [Nadi and Delavar 2011]. This study presents a generic model for personalized, multi-criteria route planning. It integrates different decision strategies, considering user preferences through pairwise comparison and quantifier-guided ordered weighted averaging (OWA) aggregation operators. The model generates multiple alternative routes with different decision strategies, allowing users to select the most suitable one based on real-world situations. Implemented in a web-based Geographic Information System (GIS) in Isfahan, Iran, the model was validated in a tourist routing scenario, demonstrating its effectiveness in practical situations.

2.2.2 [Saplıoğlu and Aydın 2018]. This study examines the positive and negative factors affecting safety and route choice in the integration of cycling with public transport, specifically focusing on Isparta City, Turkey. The study utilizes a questionnaire survey to identify key parameters, such as Accident Prone Areas (APA), Bus Lanes (BL), and Road Side Car Parks. Geographic Information Systems (GIS) and accident reports are used to determine safer routes, while the Analytic Hierarchy Process (AHP) method is applied to analyze survey data and identify effective parameters. The results emphasize the significance of Accident Prone Areas, Bus Lanes, and Road Side Car Parks in integrating cycling with the public transport system and determining safer routes. The combined use

of the questionnaire survey, GIS, and AHP proves valuable in this context.

2.3 Mobile Implementation

2.3.1 [Ibrahim and Mohsen 2014]. This paper presents an Android app enabling users to modify and assess locations on an online map. Besides standard navigation functions like route display, distance, and drive time calculation, it integrates Google Maps APIs, Google Direction APIs, PHP, JSON, and MySQL for efficient solutions. The app adopts a client/server model: the Android device as client, and PHP/MySQL as server.

2.3.2 [Fröhlich et al. 2016]. The article emphasizes precise traffic data for upcoming cities and presents BikeNow, a system suggesting cyclists adapt their speed for optimal traffic light passage. BikeNow employs Dresden's VAMOS traffic data, predicting light changes and advising speed adjustments for green lights. The article covers BikeNow's architecture, green time prediction, and user interface; implementation, including VAMOS integration; and cyclist-centric study findings. It concludes by highlighting outcomes and proposing future directions.

2.3.3 [Hu and Dai 2013]. The architecture and implementation of the online mapping application described in the paper serve as a valuable reference for developing a similar solution. The use of Google Maps API V3, Google Geocoding, Microsoft SQL Server Express database, and Spry Framework for Ajax provides an efficient and user-friendly platform for delivering digital cartographic information. The application's features, such as map types, search functions, filtering options, and real-time database updates, demonstrate the potential for creating a robust and interactive mapping solution. Overall, the described implementation serves as a useful guide for developing an effective and feature-rich mapping application.

2.4 Constrained Graph

2.4.1 [Chew 1987]. The constrained Delaunay triangulation (CDT) is a triangulation of a set of vertices in the plane that includes specified non-crossing edges and closely approximates the Delaunay triangulation. This paper demonstrates that the CDT can be constructed in optimal O(n log n) time using a divide-and-conquer technique. This time complexity matches that of building an unconstrained Delaunay triangulation and an arbitrary constrained triangulation. CDTs exhibit properties that make them useful for applications such as the finite-element method, motion planning with polygonal obstacles, and constrained Euclidean minimum spanning trees.

2.4.2 [Shewchuk 1996]. The library `Triangle®` generates meshes (i.e. planar graphs) from a set V of vertices in $\mathbb{R}^2$. One emphasis of Triangle®is the generation of Constrained Delaunay Triangulations. `Triangle®` is able to force the Triangulation Graph to respect a *set of constraint edges* that have priority over Delaunay edges. The border of a (constrained) Delaunay Triangulation equals the border of the Convex Hull of the set of vertices ($\partial DT(V) = \partial H(V)$). If V is considered to be the set of vertices of the urban bikepath system, $DT(V)$ is a fully connected graph covering V. Therefore, $DT(V)$ is a prescription for full connection among the bike system sites (i.e. bike dispensers and bikepaths). If the constrained edges input to Triangle®are the ones of the bikepaths C_i, then Triangle®would provide a fully bikepath-connected graph, while at the same time respecting the bikepaths prescribed by the city authorities. Unfortunately, due to copyright and legal circumstances, we cannot use Triangle®. However, we have devised heuristics to achieve full connectivity among the bikepath system nodes V while at the same time respecting the edges of bikepaths C_i, without using Triangle®. Such heuristics are discussed in the Methodology section.

2.5 Delaunay Triangulation in path planning

2.5.1 [Yan et al. 2008]. The paper proposes a path planning algorithm that uses a constrained Delaunay triangulation (CDT) to find an optimal path from a start point to an end point in an obstacle-filled environment. The algorithm first constructs a CDT of the environment, then constructs a dual graph on the CDT, and finally uses searching algorithms to find a path from the start point to the end point. The algorithm is simple and effective, and it is a good choice for applications where real-time performance is important.

2.5.2 [Hahmann et al. 2018]. The paper compares the performance of six algorithms for routing through polygon-defined open spaces. One of them, Delaunay, constructs a triangulation of the open space, which can be used to find paths by following its edges. Results demonstrate Delaunay's efficiency in route computation time.

2.5.3 [Sakthivel et al. 2022]. The article highlights varied uses of autonomous drones and UAVs (Unmanned Aerial Vehicles) in civil applications like agriculture, environmental protection, public safety, and traffic management. It discusses challenges and opportunities for incorporating these systems into smart city frameworks, stressing efficiency and cost-effectiveness. The study concentrates on enhancing UAV route planning, presenting an algorithmic solution employing Delaunay triangulation to reduce collision risks. Experimental outcomes underscore the effectiveness of the Lin-Kernighan technique within this Delaunay-based method, providing insights for real-world UAV deployment in intricate urban settings.

2.6 Conclusions of the Literature Review

The literature review has revealed important findings regarding bike route planning and implementation. Commercial systems, such as OpenStreetMap (OSM), offer valuable data for assessing bicycling infrastructure, although discrepancies may arise when compared to municipal open data. For relatively small cities like Medellin, these discrepancies are expected to be greater. Therefore, to ensure more accurate and reliable solutions, it is crucial to have control over the municipal data and prioritize its usage in the development of the bike route planner. By relying on authoritative and up-to-date municipal data, we can enhance the effectiveness and precision of the EnCicla®system in Medellin. Personalized route planning techniques, incorporating edge types and weighted routing, allow for multi-criteria decision-making and the generation of alternative routes based on user preferences and real-world considerations. Integrating cycling with public transport systems necessitates careful analysis of safety and route selection factors, such as Accident Prone Areas, Bus Lanes, and Road Side Car Parks. Web implementation utilizing technologies like Google Maps APIs has shown

promise in providing efficient and user-friendly navigation solutions. Additionally, constrained graph methods, like the constrained Delaunay triangulation (CDT) and alternative heuristics, enable the generation of planar graphs with specific edge constraints. These insights, along with prioritizing municipal data control, will inform the development of an effective bike route planner using the EnCicla®system in Medellin. Moreover, the application of Delaunay triangulation as a natural option for path planning is relevant. While it is predominantly employed in open space routing, its consistency and utility manifests in our context, which involves connecting non-connected subgraphs.

3 METHODOLOGY

Fig. 1 presents the main workflow of our implementation, as follows: (1) the piecewise linear (PL) bikepaths C_i click-sampled by a human are corrected to remove defects (Fig. 3). The result is a set of bikepaths without topological errors, that are promoted to a graph $G_C = (V_C, E_C)$ structure. (2) The bike dispenser sites D are added to the G_C graph. The bridge edges which add those bike dispensers represent minimal distances $d()$ to neighboring bikepaths C_i (Fig. 4(a)). The new vertex set is $G_B = G_C \cup D$. (3) The disconnected set of bikepaths C_i in G_B is connected via a Delaunay Triangulation (DT) of vertices V_B. The concept of distance $d()$ for this DT may be Euclidean, City Block, or other, relevant for the particular city at hand. (4) Non-realistic (e.g. convex hull or very long) edges of $DT(V_B)$ must be removed, as they represent impossible, risky or inconvenient bike routes for that particular municipality. (5) The fully connected graph G is used to compute minimal cost trajectories between pairs on vertices. The present client metropolitan authority requests computing of minimal cost routes among bike dispenser nodes (D) only, thus allowing a full pre-processing of the answer to user interrogations. (6) The matrix P containing the minimal-cost paths from dispenser v_0 to v_f is accessed in constant-cost manner, returning the sequence $[v_0, v_1, ..., v_f]$ of vertices.

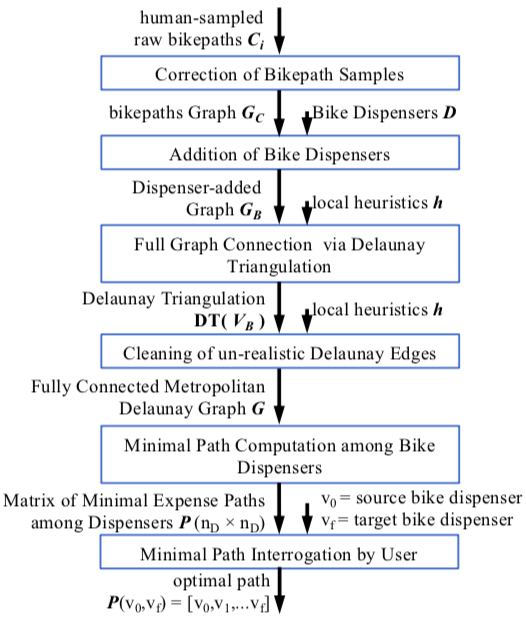


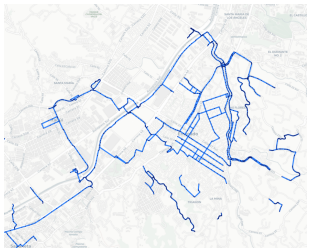
Figure 1: Main Workflow of Computing for Minimal Bike Trajectories.

3.1 Available Data

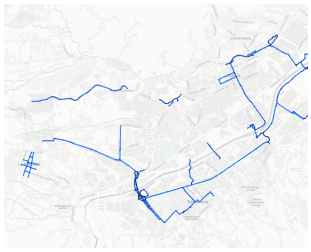
(1) The point sample to digitize the PL bikepaths $C_i(u)$ is executed by a human user via mouse clicks in a quite rough manner.
(2) $C_i(u)$ are PL curves in $\mathbb{R}^2$. $C_i(u)$ = `(longitude,latitude)`.
(3) The distance function $d(p, q)$ between two domain locations $p, q \in \Omega$, is a *metric* (non-negative real-valued, symmetric, triangular inequality-compliant). The present manuscript reports using *City-Block* distance as *metric*. There is the obvious alternative of *Euclidean* distance, besides other societal-, regulation-, environmental- and urbanistic-based metrics.
(4) The point samples $C_i(u)$ have Nyqvist-Shannon [Nyquist 1928; Shannon 1948, 1949] quality in all neighborhoods, as by definition the human user visually ensures that the point sample preserves the required bikepath features. However, the sampling interval is not constant. In low curvature regions, the User applies a long sampling interval, and vice versa.
(5) Curves $C_i(u)$ are graphs themselves in which (a) most curve-internal vertices have degree 2, (b) most terminal vertices have degree 1, and, (c) a small number of internal vertices have degree larger than 2 (i.e. where a `T` or `X` junction occurs).
(6) The set of all bikepaths (Fig. 2) $C_i(u)$ forms the graph $G_C = (V_C, E_C)$. Therefore, $E_i = (v_k, v_j)$ is an edge of E_C if $v_k = predecessor(v_j)$ or $v_k = successor(v_j)$ in a bikepath $C_i(u)$.
(7) Some bikepath curves $C_i(u)$ may look self-intersecting in $\mathbb{R}^2$, while in $\mathbb{R}^3$ they represent fully legal overpass trajectories (e.g. C_w in Fig. 3(a)). In such cases, it is important that the apparent cross of C_w against itself generate no explicit vertex in the crossing. In this manner, the apparent non-manifoldness in $\mathbb{R}^2$ is not translated into the graph topology.

3.2 Graph Construction

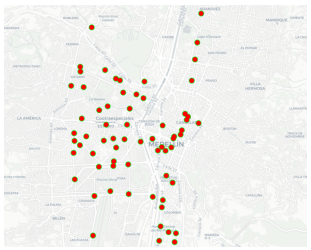
(1) **Vertex Displacement / Merging / Deletion.** Fig. 3(a) shows that in the raw input data, bikepaths C_i , C_j and C_k are topologically independent, while they may geometrically touch or intersect. If the city regulation prescribe that these bikepaths be integrated, a common vertex must be created in C_i / C_j and in C_i / C_k. This is the solution depicted in Fig 3(b). The converse situation might appear, forcing to separate bikepaths which, due to sampling noise, mistakenly share vertices .
These Figures also show bikepath C_w, which is apparently mistaken as it self-intersects in $\mathbb{R}^2$. However, it is completely correct in $\mathbb{R}^3$. All what is needed in this case is to ensure that C_w does not form a loop by re-visiting a vertex (i.e. as illustrated). Therefore, no correction is needed.
(2) **Addition of Bike Dispenser Sites to G_C.** Enlargement of the bikepath graph $G_C = (V_C, E_C)$ to include the bike dispenser sites D entails the obvious step of enlarging the vertex



(a) Borough 1. Bikepaths C_i / Graph G_C (i.e. without bike dispensers).



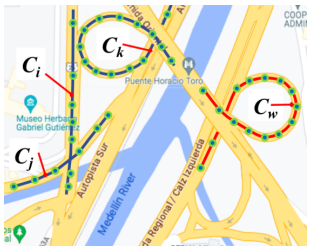
(b) Borough 2. Bikepaths C_i / Graph G_C (i.e. without bike dispensers).



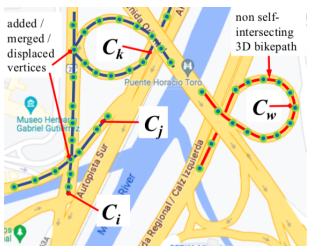
(c) Added Bike Dispenser vertices.

(d) Graph G_B (i.e. with bike dispensers and connecting edges).

Figure 2: Bikepaths C_i, bikepath Graph G_B, bike dispensers D and bikepath Graph with bike dispensers G_B.



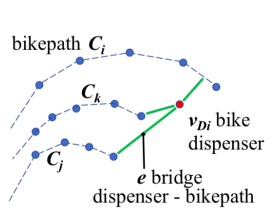
(a) **Blue bikepaths** C_i, C_j**:** topologically disconnected due to sampling errors. **Red bikepath** C_w**:** Apparent self-intersection in $\mathbb{R}^2$.



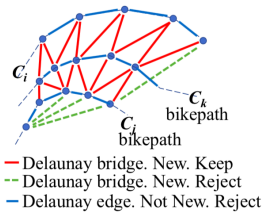
(b) **Blue bikepaths** C_i, C_j**:** topologically connected after vertex addition / merge / displacement. **Red bikepath** C_w**:** No actual self-intersection in $\mathbb{R}^3$. No correction needed.

Figure 3: Bikepath Crossings and self-crossings. Vertex Correction.

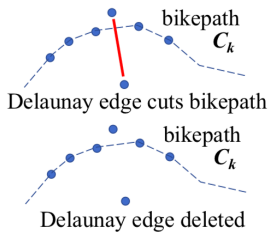
set: $V_B = V_C \cup D$. However, the generation of bridge edges from the bike dispensers to the bikepaths is the actual graph enlargement. Alternatives to add the bike dispenser sites D to the basic bikepah graph $G_C = (V_C, E_C)$ include: (i) Explicit computing of shortest vertex-to-curve paths which connect each bike dispenser v_{Di} to several bikepaths C_i (Fig. 4(a)), (ii) a parsimonious strategy of differing the computing of dispenser-bikepaths bridges to a later stage which connects all available graph vertices via a Delaunay Triangulation. Our implementation executes option (i) above.



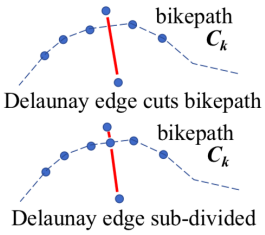
(a) DT Bridges Dispenser / Bikepath.



(b) Treatment of DT Bridges



(c) Complete priority for bikepaths C_i.



(d) Partial priority for bikepaths C_i.

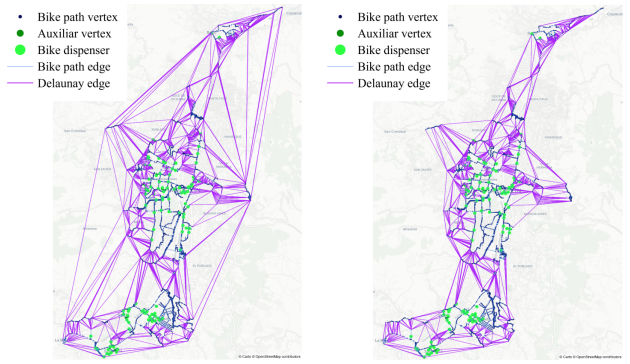
Figure 4: Possible Scenarios of Bridges. (a) Among bike dispensers v_{Di} and bikepaths C_w. (b) Provided by Delaunay Triangulation. (c),(d) Full and partial priority for bikepaths C_i.

1: **procedure** $[G_B]$=DispenserInsertion(D,G_C) ▷ Add Bike Dispensers
2: ▷ G_C= Piecewise Linear sampled bike Pathways, D= Bike dispenser sites
3: **for** each v_b `bike dispenser in` D **do**
4: Bridges = ShortestBrigdes(v_b, G_C)
5: **for** each b `bridge in Bridges` **do**
6: **if** (|b| ≤ `BridgeThreshold`) AND (interior(b) ∩ $G_C = \Phi$) **then**
7: $G_C = G_C + edge(b)$
8: **end if**
9: **end for**
10: **end for**
11: $G_B = G_C$ ▷ G_B= Pathway graph with bike dispensers
12: **end procedure**

3.3 Constrained Delaunay Graph

In reference to Fig. 5:

(1) The bikepaths C_i are mostly disjoint from each other.
(2) Connection among bikepaths C_i is required, in order to make accesible all Ω urban area.
(3) A Delaunay triangulation on the vertices of $G = (V_B)$ is a natural choice for full connectivity among the bikepaths C_i.
(4) In a basic Delaunay Triangulation, the edges of the bikepaths C_i are not taken into consideration. Delaunay produces its own edge set.
(5) To enforce the bikepaths C_i, *Constrained Delaunay Graph* is achieved by preferring an edge of C_i whenever it collides with a Delaunay edge. The resulting graph G *is not a triangulation* anymore.



(a) Constrained Delaunay Graph of Bikepath Vertices.

(b) Pruned Constrained Delaunay Graph of Bikepath Vertices.

Figure 5: Constrained Delaunay Graph of the Bikepath Vertices. Basic and Pruned versions.

(6) Delaunay outermost edges form a convex hull H for vertices V_B. These edges circumvallate G and form the border of H, ∂H.
(7) Edges $e \in \partial H$ are not realistic. For example, they may join neighboring urban-surrounding cusps, as a hypothetical *teleferic* cable would. Therefore, these edges $e \in \partial H$ are deleted.

1: **procedure** $[G]$=FullGraphConnection(G_B) ▷ Fully Connects Bike route Graph
2: ▷ $G_B = (V_B, E_B)$= Set of disjoint Pathway graphs (incl. bike dispensers)
3: T_D= DelaunayTriangulation(V_B, LocalHeuristics)
4: **for** each e Delaunay Edge in T_D **do**
5: **if** (e is irreal) **then**
6: $T_D = T_D - \{e\}$
7: **else**
8: e = Prune(e, LocalHeuristics)
9: **end if**
10: **end for**
11: $T_D = T_D + E_B$ ▷ Make Pathways E_B compulsory
12: $G = T_D$ ▷ G= Fully Connected Bike route Graph
13: **end procedure**

3.4 Graph Costs Assignment

(1) Delaunay Triangulation is produced on vertices V_B (bike route plus dispenser vertices), using variate metrics $d()$.
(2) A Metric $d() : \Omega \rightarrow \mathbb{R}$ must satisfy:
  (a) $d(v_i, v_j) \geq 0$ for all v_i, v_j
  (b) $d(v_i, v_j) = 0 \leftrightarrow v_i \equiv v_j$
  (c) $d(v_i, v_j) = d(v_j, v_i)$ for all vertices v_i, v_j
  (d) $d(v_i, v_j) \leq d(v_i, v_k) + d(v_k, v_j)$ for all vertices v_i, v_j
(3) Urban Metrics $d()$ include: (a) Euclidean, (b) Street-block, (c) Safety-biased, (d) Path Quality-biased. However, other metrics are usable.
(4) Observe that cost functions such as "slope difficulty" are not commutative and therefore they are not currently used as graph cost metrics.
(5) Notice that when the graph is directed, the cost $d(v_i, v_j)$ differs from $d(v_j, v_i)$. This case happens, for example, in heavy slope terrain. Our implementation also accepts this type of cost structure

3.5 Heuristics

3.5.1 Urban Heuristics. The following are examples of Urban Heuristics used in our implementation:

(1) **Usage Priorities. Bikepaths vs. Car-bike Shared roads.** The addition of the bike dispensers D to the basic bikepaths G_C implies the possible rejection or fragmentation of dispenser-bikepath bridges which intersect other bridges or bikepaths. For example, if the city authorities want to prioritize usage of bikepaths over car-bike shared roads, then priority is given to reach a bikepath as soon as possible even if an additional route car-bike shared roads would result in a more economic overall trajectory.
(2) **Pruning of Delaunay Triangulation Edges.** Fig. 5(a) shows the basic Delaunay Triangulation $DT(V_B) = DT(V_C \cup D)$. This figure also illustrates that usage of long Delaunay edges defeats the intention of using bikepaths. Also, those Delaunay edges may be unrealistic. For example, the border edges ($\partial DT(V_B)$) are a circumvallation of the urban area Ω. However, in most cities, the existence of such a road is an urbanistic utopia, or is risky due to terrain, criminality, or other conditions. Therefore, such Delaunay edges must be deleted.

3.5.2 Geometric Heuristics. The following are the main Geometric Heuristics used in our implementation:

(1) **Full Connectivity of the G Graph vs. Prunning of Delaunay Triangulation of V_B.** The Delaunay Triangulation $DT = DT(V_B)$ ensures full connectivity of the graph G (bikepaths plus bike dispensers). However, is pruned to eliminate bike routes in the city domain Ω which may be overly long or simply un-realistic (e.g. a path joining two mountain tops). The pruning of DT lowers its connectivity and might defeat the full connectivity goal for G=`pruning(DT)`. However, this disconnection has not appeared in the trajectories delivered to the public.
(2) **2D Delaunay Triangulation of V_B applied on bikepaths / roads with overpasses 3D.** As illustrated in bikepath C_w of Fig. 3, the 2D projection of this 3D bikepath does not deform the graph G as long as the self intersection of the 2D projection of C_w is not added as a vertex to C_w. However, when $DT(V_B)$ is computed, an edge of $DT(V_B)$ may exist, which joins a vertex at ground level with another vertex at overpass level, being therefore an unreal bridge. Our algorithm does not, at this time, prevent this result. However, this malfunction has not occurred up to the present time.
(3) **Delaunay Edges as Shortcuts of Bikepaths C_i.** A Delaunay Triangulation edge may join two vertices of the same bikepath C_i. In this case, such a DT edge is ignored, as it

subverts the priorities of the municipalities for the citizens to use bikepaths C_i whenever available.

(4) **Priority of Bikepaths C_i over Delaunay Edges.** Whenever a Bikepath C_i cuts a Delaunay edge, the Delauney edge is either (i) eliminated (Fig. 4(c)) or (ii) sub-divided by the bikepath, as per municipality priorities (Fig. 4(d)). If the urban regulations prescribe as absolute priority the usage of the bikepaths C_i, option (i) is chosen. Otherwise, a portion of the Delaunay edge may be kept as a manner to access the bikepaths C_i with several options. The results presented in this manuscript correspond to absolute priority of bikepaths C_i over Delaunay edges (i.e. option (i) above).

3.6 Pre-Processing

(1) Data Pre-processing: Cleaning of $C_i(u)$ bikepaths. Initial vertices are sampled by a human user via a Graphic User Interface, thus presenting digitization errors. These errors manifest themselves, for example, when two vertices (of different curves) should reside in the same $\mathbb{R}^2$ point and yet they are slightly off from each other. This defect, in turn, would obviously derail the connectivity tests among bikepaths C_i.

(2) Data Pre-processing: Two bikepaths C_i and C_j may cross each other with or without actual intersection. If C_i crosses over C_j (e.g. via a bridge), these bikepaths do not intersect. From a graph viewpoint, C_i and C_j are independent. If C_i does intersect C_j, it is compulsory to create a vertex, common to C_i and C_j. It would be incorrect to allow this intersection to remain implicit, via intersection of the interiors of edges: e.g., $int(ab) \cap int(cd)$.

(3) Interrogation Pre-processing: In the present effort, the client metropolitan authority has demanded to offer the public the planning of routes among bike dispenser nodes in D ($O(|D|) \approx 100$). The client does not require, at his time, computing the optimal path among any pair of vertices of the set V_B ($O(|V_B|) \approx 7000$). This specification offers the opportunity to pre-compute and store the optimal path from dispenser v_{Di} to dispenser v_{Dj} for all combinations of the origin and destination bike dispensers.

3.7 Mobile Implementation

The solution bike rides from dispenser D_i to dispenser D_j is stored in matrix R. Each $R(i, j)$ contains (a) the solution bike ride, (b) its cost, (c) the corresponding walk and (d) the walk cost. If the edge cost corresponds to a *metric*, then an edge expense e_{ij} equals e_{ji}. It follows that $d(v_i, v_j) = d(v_j, v_i)$ and we may assume that R is symmetric. Therefore, only half of R must be stored. The diagonal of R is also not needed since it would register bike rides from a dispenser to itself. As a result, R represents $\frac{n_D(n_D-1)}{2}$ cells. The coordinate sequences for the solution bike ride and for the walk are compressed and stored by Google Maps®[Google nd]. In this particular scenario, the matrix R (compressed bike ride, walk path and their costs) is very large. Thus, it is impractical to store it in the client device. As a consequence, this matrix is stored in the EnCicla®database (i.e., server of the municipal authority)."

The mobile app for the EnCicla® public bike system (Fig. 7) follows a client-server architecture (Fig. 6). The **client** component is responsible for handling user requests, interacting with the Google

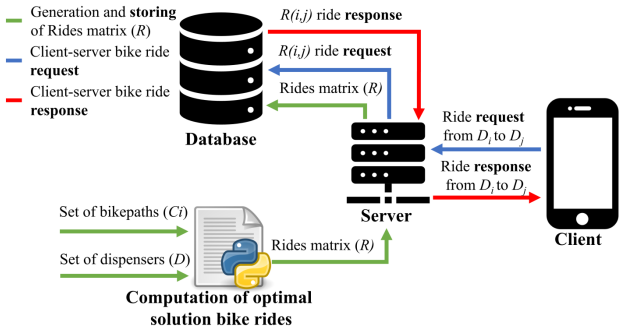


Figure 6: Data flow in a client-server app architecture: Calculating, storing, and requesting the matrix of optimal solutions.

Maps™ service when necessary, and displaying the requested solution ride. On the other hand, the **server** component handles client-app requests, including user data, dispenser availability, and, most importantly, retrieval of the optimal solution bike rides from the solution matrix stored in the **database**. It is important to notice that the the **optimal solution bike rides** is computed once, in a time previous to user interaction. The resulting solution matrix is uploaded to the database, which the server accesses when the app requests it.

In this web-based setup, a significant method resembling a digital twin emerges. Using a Gaussian bell curve alongside historical records data, we can estimate bike availability in dispensers. This approach effectively mirrors real-world processes by digitally representing system dynamics. It predicts bike availability based on historical trends, synchronizing data insights with practical outcomes.

It is important to mention that the app is actually Web-compatible, but the commercial client did not required it to be web deployed.

3.7.1 *Color and style of Solution Bike Rides in Mobile App:*

(1) Dark blue: Bikepaths excluding cars.
(2) Light blue: Car & Bike shared paths. Road paths with priority given to bikes.
(3) Yellow: Delaunay edges or Bridges between Dispenser and Bikepaths C_i.
(4) Black (dotted line): Real-time Google Maps™ path connecting an arbitrary point to the nearest dispenser (Fig. 7(b)).

4 RESULTS

4.1 Resulting Graph

Figure 8 depicts a portion of the resulting fully connected, non-directed, and weighted Graph G, along with its corresponding edge types, as follows:

(1) Bikepath edge: Edges of C_i which are exclusively designated for bikes.
(2) Car & Bike shared edge: These edges are shared with cars but give priority to bikes.
(3) Connecting edges: Connect bike dispensers to the bikepaths C_i.

(a) Let v_{orig} and v_{dest} be two points such that $v_{\text{orig}}, v_{\text{dest}} \in \Omega$ and $v_{\text{orig}}, v_{\text{dest}} \in D$.

(b) Let v_{orig} and v_{dest} be two points such that $v_{\text{orig}}, v_{\text{dest}} \in \Omega$ and $v_{\text{orig}}, v_{\text{dest}} \notin D$.

Figure 7: Implementation of solution rides in mobile app. Route between origin v_{orig} and destination v_{dest} vertices.

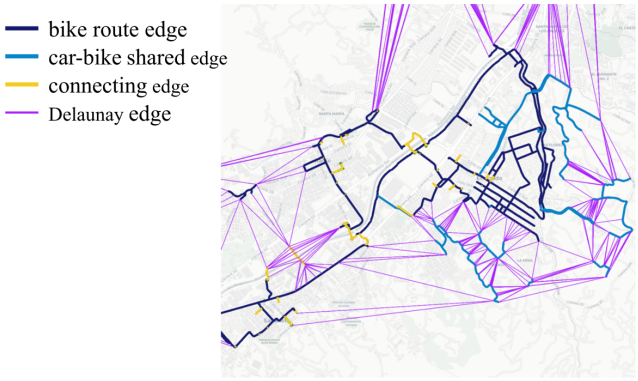


Figure 8: Fully connected urban bike route Graph G. Edge types.

(4) Delaunay edge: Connect Bikepaths C_k and C_w or connect Dispensers to Bikepaths C_i.

Fig. 7 displays Delaunay edges as straight segments. However, they have City-block pattern.

4.2 Efficiency Ratio

The Efficiency Ratio ER (relevant values in Table 1) for our application divides the cost $d(v_{Di}, v_{Dj})$ of the solution bike ride (between dispensers) over the cost $d_w(v_{Di}, v_{Dj})$ of the corresponding walk (computed by Google Maps™).

$$\text{ER} = \frac{d(v_{Di}, v_{Dj})}{d_w(v_{Di}, v_{Dj})} \tag{1}$$

We do not have the resource capacity to obtain the licenses of commercial software and to populate those data bases with the proprietary data. We do not have the legal permission of the client to store its data in other systems. In addition, the composition of the bike rides is completely different in each one of the commercial systems, thus leading to non-comparability of scenarios.

However, we were able to contrast the quality of the computed bike rides against their pedestrian walk counterparts. It turns out that the triangulation-derived bike rides tend to be almost identical

Table 1: Bike Ride Efficiency Indicator 1: Comparison of Bike-to-Walking Distance Ratio for the Metropolitan Area Ω.

Bike Ride Efficiency Indicator	Value	Observations
Mean value	1.19	ER= 1.19 indicates that, on average, the solution bike Ride provides a favorable alternative compared to walking routes.
Standard Deviation	0.17	The Efficiency Ratio values exhibit a relatively low standard deviation of 0.17, suggesting a relatively narrow range of variability around the mean value.
Minimum value	0.31	Cases where $d(v_{Di}, v_{Dj}) < d_w(v_{Di}, v_{Dj})$, indicating instances where the bike rides significantly reduce the distance between dispenser nodes compared to walking routes.
Maximum value	3.65	Cases where $d(v_{Di}, v_{Dj}) > d_w(v_{Di}, v_{Dj})$, indicating instances where the bike rides are considerably longer in distance between dispenser nodes compared to walking routes.

to the ones naturally walked by a pedestrian in terms of distance cost (Table 1). That is, the mean value of the Bike Ride Efficiency Indicator is 1.19 (very near the ideal value 1.0).

Note: Our reported implementation is the result of the requirements placed by the Metropolitan authority. One of the *sine qua non* conditions was that property of the GIS data would remain with the municipality instead of being owned by foreign IT companies.

4.3 Complexity Analysis

Consider n_C=total number of vertices of the bikepaths, n_D=number of bike dispensers. Table 2 presents the time complexities of the diverse tasks entailed by EnCicla ®. This Table displays the expenses in $O()$ notation, for the different processes involved in both (a) the graph construction and (b) the interrogation of minimal-cost bike trajectories. Notice that an additional improvement is possible, since our need of minimal paths only involves dispenser nodes $v_D \in D$ and not all graph nodes $v_i \in V_B$. Therefore, the standard Dijsktra used is somehow over-dimensioned for the present needs.

4.4 Expenses of Web Services

The manuscript presents a detailed cost analysis of integrating the Google Maps™ API into a routing algorithm solution. The focus of this analysis is to assess the financial implications associated with the API usage. By examining the cost structure and patterns of API calls, valuable insights are gained regarding the expenses involved in implementing the routing algorithm. This examination allows for a thorough evaluation of the economic considerations and feasibility of incorporating the Google Maps™ API into the solution. Additionally, it is important to note that there may be other associated costs when using the app (see Table 4). These findings contribute to a broader understanding of the financial aspects and practicality of utilizing external APIs in routing algorithms.

Table 2: Time Complexity Analysis. Bike dispenser set is D with $n_D = |D|$. Bikepaths sampling point set is V_C with $n_C = |V_C|$. Full vertex set is $V_B = D \cup V_C$. Notice that: $n_B \gtrapprox n_C \gg n_D$. Therefore $n_C \approx n_B$.

Process	Complexity	Observations
Bikepaths Correction: Point and Intersections Correction	$O(n_C^2)$	Correction of bikepath vertex sites and explicit generation of path-path intersections
Dispenser Addition:	$O(n_C.n_D)$	Computing of dispenser / bikepath bridges
Delaunay Triangulation	$O(n_B.log(n_B))$	DT of n_B vertices in $\mathbb{R}^2$
Purge of irreal Delaunay edges	$O(n_B^2)$	According to actual terrain conditions, city policies, etc.
Shortest Paths among all Bike Dispensers D	$O(n_B^2.n_D)$	Repeated invocation of Dijkstra [Dijkstra 1959] algorithm. In-house implementation.
TOTAL	$O(n_B^3)$	Assuming that $n_B \gg n_D$ (bikepath sampling vertex set much larger than dispenser set.)

Table 3: Cost Analysis of Google Maps™API service to compute solution. Cost per *Directions Advanced* API call: 0.01 USD [Google 2023]

.

Usage	Number of API calls	Cost (USD)	Observations
Strictly triangular distance matrix	5995	$59.95	This matrix serves analytical purposes, as it facilitates the generation of the Bike-to-Walking Distance Ratio (Table 1) and stores the walking path to be utilized within the app.
Delaunay edges cost assignment	1693	$16.93	These API calls are utilized to assign costs to Delaunay edges in section 3.4

Table 4: Cost Analysis of Google Maps™ [Google 2023] API Usage by Human Users. Expenses are presented in US cents. Approximated total cost per application usage: 12.4¢

API name	API cost	App Calls	Total cost	Notes/Comments
Dynamic maps	0.7¢	5	3.5¢	Standard cost for displaying a map, incurred whenever a user accesses the map interface within the application.
Directions Advanced	1¢	4	4¢	Provides a walking route between two points when the user's start and/or end locations don't match a dispenser node (D), requiring guidance to the nearest one. (Fig. 7(b))3.4
Autocomplete	1.7¢	2	3.4¢	Generates a list of suggested place predictions from a provided input string. Applied when a user intends to enter the origin or destination for their ride.
Geocoding	0.5¢	3	1.5¢	Returns data from a location on map. Used to get data from user manual origin/destination selection.

5 CONCLUSIONS

This manuscript has presented the mobile-based implementation of the bike urban route planner EnCicla®. This planner constructs a fully connected bike route graph taking as input a set of bikepaths C_i, digitized in Piecewise Linear manner. These bikepaths are in general disconnected from each other. Therefore, a constrained Delaunay Triangulation strategy has been used to connect the bikepaths C_i with each other. In addition, a set D of bike dispensers has been integrated to form the full urban bike accessibility graph G. The Topology and Geometry of graph G obeys both urban regulation and geometric heuristics. Thus, the graph G reflects the policies of the municipality regarding bike-based sustainable transportation. The current demands from the metropolitan authorities include the planning of bikepaths among bike dispensers. Thus, the planned bikepaths start and finish in bike dispensers $d_i \in D$. The routes include sub-trajectories in (a) bike-devoted, (b) signalized car-and-bike and (c) standard automotive roads/lanes.

The bike rides have minimal graph traversal cost. However, the cost goes beyond Euclidean or City-block, to include any expense function (e.g. pollution, safety, security, terrain difficulty) that the city authority may wish to impose.

The EnCicla®bike rides planner currently covers a metropolitan area with 10 cities/boroughs and a population of 4 million people. The number of bike dispenser customers (i.e. public bike borrowers) approaches 130.000. However, the number of actual bikers using the planner (i.e. using own or public bikes) is much larger.

6 FUTURE WORK

At the present time, the demands of the client cities imply that the formulation of an optimal bike rides can be pre-computed. This situation would change in the following scenarios:

(1) Demand for overall *point-to-point* instead of *dispenser-to-dispenser* bike rides. This demand would obviously make the processing times dramatically larger as the number of vertices would grow to roughly n_I=number of street intersections in the metropolitan area Ω.
(2) Demand for time-dependent edge costs. This scenario relates to rush-hour vehicle congestion and contamination edge expenses, which depend on the actual time of the route.
(3) Demand for non-symmetrical edge costs ($d(p, w) \neq d(q, p)$). This scenario occurs, for example, in a slope, where one of the route directions has significantly different cost in either direction. The scenario would make the graph V a *directed* one,

increasing the storage and computing resources needed to plan the trajectories.

Exploring these potential enhancements and their application in scenarios like digital twin concepts could offer valuable ways for extending the capabilities and impact of the EnCicla®bike rides planner.

ACKNOWLEDGMENTS

This research was conducted with the resources from Cohesive Manufacturing, the Metropolitan Area of Valle de Aburra, Colombia, and under the scientific supervision of the Laboratory of CADCAMCAE at EAFIT University.

- Laboratory of CAD CAM CAE website: http://www1.eafit.edu.co/cadcamcae
- Cohesive Manufacturing website: https://www.cohesivemanufacturing.com
- Metropolitan Area of Valle de Aburra website: https://www.metropol.gov.co

REFERENCES

Rudy Ariyanto, Erfan Rohadi, and Annisa Puspa Kirana. 2022. Implementing A Star for Bicycle Route Finding System using OSM and GraphHopper: Case Study: Batu, Indonesia. In *2022 International Conference on Electrical and Information Technology (IEIT)*. IEEE, New York, NY, USA, 307–312. https://doi.org/10.1109/IEIT56384.2022.9967899

L Paul Chew. 1987. Constrained Delaunay triangulations. In *Proceedings of the third annual symposium on Computational geometry - SCG '87*. ACM New York, NY, USA, Waterloo Ontario Canada, 215–222. https://doi.org/10.1145/41958.41981

Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

Colin Ferster, Jaimy Fischer, Kevin Manaugh, Trisalyn Nelson, and Meghan Winters. 2020. Using OpenStreetMap to inventory bicycle infrastructure: A comparison with open data from cities. *International journal of sustainable transportation* 14, 1 (2020), 64–73.

Sven Fröhlich, Thomas Springer, Stephan Dinter, Sebastian Pape, Alexander Schill, and Jürgen Krimmling. 2016. BikeNow: A Pervasive Application for Crowdsourcing Bicycle Traffic Data. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (Heidelberg, Germany) *(UbiComp '16)*. Association for Computing Machinery, New York, NY, USA, 1408–1417. https://doi.org/10.1145/2968219.2968419

Google. 2023. Directions API Billing Cost for Advanced Searches. https://developers.google.com/maps/billing-and-pricing/pricing

Google. n.d.. Encoded Polyline Algorithm Format. https://developers.google.com/maps/documentation/utilities/polylinealgorithm

Stefan Hahmann, Jakob Miksch, Bernd Resch, Johannes Lauer, and Alexander Zipf. 2018. Routing through open spaces – A performance comparison of algorithms. *Geo-spatial Information Science* 21, 3 (2018), 247–256. https://doi.org/10.1080/10095020.2017.1399675 arXiv:https://doi.org/10.1080/10095020.2017.1399675

Shunfu Hu and Ting Dai. 2013. Online map application development using Google Maps API, SQL database, and ASP .NET. *International Journal of Information and Communication Technology Research* 3, 3 (2013), 1–10.

Omar A Ibrahim and Khalid J Mohsen. 2014. Design and implementation an online location based services using Google maps for android mobile. *International Journal of Computer Networks and Communications Security (CNCS)* 2, 3 (2014), 113–118.

Saeed Nadi and Mahmood Reza Delavar. 2011. Multi-criteria, personalized route planning using quantifier-guided ordered weighted averaging operators. *International Journal of Applied Earth Observation and Geoinformation* 13, 3 (2011), 322–335.

H. Nyquist. 1928. Certain Topics in Telegraph Transmission Theory. *Transactions of AIEE* 47 (1928), 617–644.

M. Sakthivel, Shashi Kant Gupta, Dimitrios A. Karras, Alex Khang, Chandra Kumar Dixit, and Bhadrappa Haralayya. 2022. Solving Vehicle Routing Problem for Intelligent Systems using Delaunay Triangulation. In *2022 International Conference on Knowledge Engineering and Communication Systems (ICKES)*. IEEE, Udaipur, India, 1–5. https://doi.org/10.1109/ICKECS56523.2022.10060807

M Saplıoğlu and MM Aydın. 2018. Choosing safe and suitable bicycle routes to integrate cycling and public transport systems. *Journal of Transport & Health* 10 (2018), 236–252.

C.E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 379-423 (1948), 623–656.

C.E. Shannon. 1949. Communication in the Presence of Noise. *Proceedings of the IRE* 37, 1 (1949), 10–21. https://doi.org/10.1109/JRPROC.1949.232969

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering*, Ming C. Lin and Dinesh Manocha (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 203–222.

Hongyang Yan, Huifang Wang, Yangzhou Chen, and Guiping Dai. 2008. Path planning based on Constrained Delaunay Triangulation. In *2008 7th World Congress on Intelligent Control and Automation*. IEEE, Chongqing, China, 5168–5173. https://doi.org/10.1109/WCICA.2008.4593771