ALGORITHM FOR PLANNING FASTER ROUTES IN URBAN NETWORKS WITH TIME-DEPENDENT ARCS AND THE POSSIBILITY OF INTRODUCING WAITING PERIODS AT NODES

FRANCISCO A. ORTEGA, GUIDO MARSEGLIA, JUAN A. MESA & RAMÓN PIEDRA-DE-LA-CUADRA
Mathematics Institute (IMUS), Universidad de Sevilla, Spain

ABSTRACT

Navigation systems implemented in mobile devices allow users to search for the shortest routes between pairs of points. Many of the existing commercial products assume in a simplified way that the travel time to cross each arc of a road network is fixed, once a starting time has been established. However, the real travel time along a road section within cities depends on many factors that are related to traffic congestion, weather conditions, possible incidents, etc., and consequently, it depends on the time. As can easily be shown, determining the shortest itineraries in a network whose arcs are time-dependent can result in a diversity of optimal routes for a same origin–destination pair based on different departure times. Assuming the availability of the estimated data of the time required to travel along each section of the street network, once the departure time has been previously set, we propose in this work an efficient algorithm for obtaining faster routes on time-dependent arcs, in such a way that the sum of driving times is minimized, which in parallel allows improving fuel consumption and reducing associated polluting emissions. The possibility of introducing waiting periods in the nodes to optimize the total time spent on the trip has also been considered in the design of the proposed procedure. An experimental evaluation is carried out to show the effectiveness of the provided algorithm.

Keywords: route planning, guided transport systems, mobility, transport networks, Dijkstra's algorithm.

1 INTRODUCTION

From a theoretical point of view, the problem of finding a shortest path from one node to another in a graph with fixed lengths (or fixed travel times) on its arcs is satisfactorily solved. In route planning problems for a single objective, Dijkstra's algorithm (Dijkstra [1]) is preferably used to obtain the solution to the problem of determining the shortest path between two nodes of a connected graph. Its algorithmic complexity is $O(m + n \log n)$, where n is the number of nodes and m the number of arcs of the underlying graph (Cormen et al. [2]).

However, this algorithm in its simplest form can be impractical in many real application scenarios. One of such situations arises when the travel time between arcs is depending on the traffic intensity existing between their respective starting and ending nodes at certain period, which is known as time-dependent arcs. This circumstance can occur both in the determination of optimal public transport routes (bus, metro or dense networks of commuter train) and in private transport routes (own motorized vehicle, bicycle or similar). Therefore, it arouses great practical interest.

As illustration, Fig. 1 shows the evolution of the average intensity in the city of Seville (Spain) throughout the current day (red) compared to the evolution of yesterday (blue), expressed in number of vehicles per lane every 5 minutes. This figure also shows the occupancy levels (green), both today and yesterday, which empirically shows the existence of a correlation between both variables and explains the daily evolution of the circulating difficulty.

WIT Transactions on The Built Environment, Vol 212, © 2022 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/UMT220031

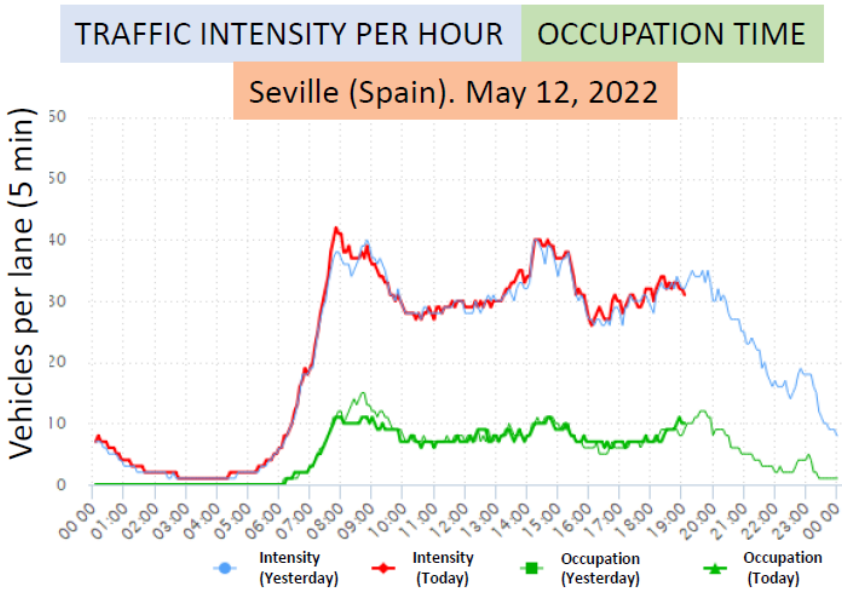

Figure 1: Evolution of the average intensity in the city of Seville along a day. *(Source: http://trafico.sevilla.org/.)*

In general, routing is the process of selecting the "best" paths in a graph G = (V, A), where V is a set of nodes and A is a set of directed arcs. Most studies on routing problems have been carried out under the assumption that all the information needed to formulate such routing problems is time invariant (Toth and Vigo [3]). In many practical applications, this assumption is generally not satisfied since travel times can vary exogenously due to traffic congestion, weather conditions, etc., or endogenously, depending on the decisions freely adopted by the driver, like modifying the speed of the vehicle at its discretion (for example, to adjust fuel consumption) or altering the travel time by including rest periods in driving. A classification of time-dependent routing problems with respect to various criteria can be seen in Pillac et al. [4].

Planning routes for vehicles in general on geographical maps, at a city, region or country scale, is a problem of great practical interest (Preuss and Syrbe [5]). The route calculation offered by Google Maps allows users to know an estimated time of arrival at the selected destination along the itinerary. We can assume that this calculation is made using the maximum allowed speed (1) and the recommended speed in each section (2) as input data. These values, in turn, could be averaged considering the intensity of traffic in real time (3), as well as historical data recorded by other users on previous occasions, if available (4). For a central server that had to respond to a potentially very large number of online requests from clients through a web interface, it would be desirable that the maximum response time of the system, providing a good solution, be limited in a way reasonable (Delling and Wagner [6]). In any case, the procedure followed to obtain these predictions offered by commercial web applications is subject to business confidentiality.

Currently, there are multiple operators that provide information on traffic conditions based on the observed speed of movement of vehicles along the sections. However, this information is not complete since it does not cover all circulating vehicles, either because there is a "non-

connected" portion in the fleet of vehicles, or because the sources of this information do not cover all browsers, telephone operators or types of vehicles. For example, Google-Traffic and InfoTransit are two real-time traffic information servers that operate in Spain based on data from telephone operators, specific vehicle fleets (insurers, carriers, automobile clubs, etc.), fixed traffic sensors, fixed traffic and GPS navigation systems. Eglese et al. [7] examine the problems related to the construction of a database of time-dependent travel times for an application developed in England.

Time-dependent routing problems have been treated as a variant of the vehicle routing problem (VRP) for a multiplicity of contexts. Among them (see Gendreau et al. [8]):

- Route planning aircraft, ships or submarines in two-dimensional or three-dimensional space, where decisions include not only the determination of the itinerary, but also the adjustment of power. The time dependency factor can be caused by air currents, ocean swell, or underwater flow. The objective to be minimized is usually the journey time, fuel consumption (or, equivalently, CO_2 emissions) or a combination of them (Perakis and Papadakis [9] and Norstad et al. [10]).
- Other causes that may imply a dependency on the travel time for the determination of the length of an optimal route would be the need to refuel mobile units (Helvig et al. [11]), or the necessary interception of the trajectories of other vehicles (Jiang et al. [12]).
- Finally, the presence of moving obstacles could also be a determining factor when planning the robot movement for such contexts (Sutner and Maass [13], Latombe [14] and Fujimura [15]).

We will assume in this work that the objective that the user always pursues is the achievement of a minimum travel time, which means reaching the destination in the shortest possible time once the starting time for the trip has been established at the origin point. This specific perspective is not the only one that can arouse interest in the specialized literature on transport. For example, public transport users may show different personal preferences when establishing their own concept of optimality (user point of view), which would imply an adaptation of the initial algorithms to accommodate the formulation of other types of optimality. advantageous situations or penalties. Among these adaptations is the consideration of the use of algorithms that produce K-shortest paths (known as KSP problem) to obtain a reasonable number (K) of shortest feasible paths and rank them based on the incorporation of user preferences.

The preferences of public transport users are usually several. As an example:

1. Achieve a minimum travel time, which means reaching the destination in the shortest possible time from the departure time established as the start from the point of origin.
2. Minimize the number of transfers: some travellers prefer to travel in a single vehicle (bus or train), rather than endure the inconvenience of transfers, even though the overall journey time may be longer.
3. Achieving a minimum travel distance route: A traveller loaded with heavy or uncomfortable objects might prefer to walk to the nearest bus stop, as a first step, instead of traveling a greater distance to another bus stop, although this second option would mean a faster route in time.

The calculation of the fastest routes between points in very extensive road networks, whose arcs depend on time, must be carried out through the participation of centralized information services based on an accessible web, where both congestion patterns and traffic data updated in real time are managed.

2 MODELLING THE PROBLEM

Following on, the notation previously used in the work of Dreyfus [16] is now described, where the travel times required to connect a pair of adjacent nodes may be dependent on the time at which the effective travel through the arc begins.

Let the graph $G = (V, A)$ where V is a finite set of n nodes or vertices, and A is a finite set of m arcs that connect the nodes. Each arc can be denoted as the ordered pair (i, j) when it corresponds to the pair of vertices i and j. Let O (origin or initial node) and D (destination or terminal vertex) be two given nodes of G; a path p from O to D in G is defined as the alternating sequence of vertices and arcs: $p = \{O = v_0, a_1, v_1, a_2, v_2, \dots, a_k, v_k = D\}$, such that:

- $a_i \in A, \forall i = 1, \dots k;\ v_i \in V,\ \forall i = 1, \dots k-1.$
- $a_i = (v_{i-1}, v_i) \in A, \forall i = 1, \dots k.$
- $O, D \in V \backslash \{v_1, v_2, \dots, v_{k-1}\}.$

The cost associated with arc (i, j) is defined as a positive real number c_{ij}. In this way, the cost of a path p will be equal to the accumulated sum of the costs of the arcs that compose it:

$$c(p) = \sum_{(i,j)\in p} c_{ij}. \tag{1}$$

Let P_{ij} be the set of all paths from i to j in the graph G. Following the notation of Dreyfus [16], the problem of finding the fastest path between points O and D where the travel time between point i and point j depends on the starting time of point j can be formulated as follows. Let us denote by $f_i(t)$ the minimum invested travel time until reaching destination D starting from point i at time t. Moreover, let $d_{ij}(t)$ be a positive value that represents the travel time invested when travelling arc (i, j) starting from point i at time t.

The recurring scheme:

$$\begin{cases} f_i\,(t) = \min\limits_{j \neq i}\ [d_{ij}\,(t)\ +\ f_j\,(t\ +\ d_{ij}\,(t))] \\ \qquad\qquad f_D\,(t)\ =\ 0, \end{cases} \tag{2}$$

allows us to design an iterative algorithm that, based on the procedure devised by Dijkstra [1], provides the fastest route, in this context of travel times in time-dependent arcs, to obtain $f_O\,(t)$.

Originally, Dijkstra's algorithm is designed to solve the problem of obtaining optimal paths in networks with not time-dependent arcs. An encoding of this algorithm is presented below.

Dijkstra's algorithm (not time-dependent arcs)

1. **Read** graph **G= (V, A),** adjacency matrix **Ady** and matrix **D** of displacement costs between pairs of adjacent points.
2. **Build** vector **f** that will store the minimum displacement costs to the origin from each node of **G** (initially, only the nodes adjacent to the origin will not have infinite value)**.**
3. **Build** vector **p** that stores the successor nodes to each node of G in the optimal path (initially, only nodes adjacent to the origin can be determined).
4. **Initialize** set **S** of nodes explored by including only the origin point. **Initialize** set **LIST=G\S.**
5. **While** LIST is not empty:
   - **5.1 Identify** index **j***

     $j^* := Arg\left[\min\{f(j) : j \in LIST\}\right]$

5.2 Remove index j* from LIST

5.3 For each successor **k** of j* included in LIST:

If $f(k) > f(j^*) + d_{j^*k}(0)$ **then**

a. **Update** $f(k) := f(j^*) + d_{j^*k}(0)$

b. **Update** $p(k) := j^*$

6. **End**.

The following two figures graphically reproduce the same example used in Wen et al. [17] to illustrate the operation of Dijkstra's algorithm and its limitations when the network is time dependent. In Fig. 2 a static situation is considered, where the weights of the arcs do not vary with time.

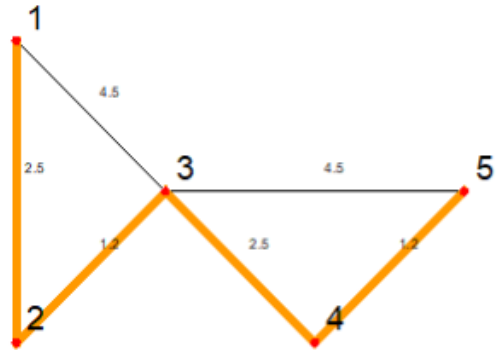

Figure 2: Quickest path between nodes 1 and 5 using Dijkstra's algorithm.

As illustrated in Fig. 2, the solution to the problem is the path 1-2-3-4-5, and the time spent is 7.4 (2.5 + 1.2 + 2.5 + 1.2) minutes, as can be verified by applying the step-by-step algorithm. Let now assume that arc (3,5) is time-dependent in the following way:

1. When the trip along arc (3,5) is started at node 3 in a time t included in the interval [0,4], the total time needed to traverse arc (3,5) is 4.5 minutes, as in the previous static case.
2. However, when the trip through the arc (3,5) starts after time 4, then the travel time needed to traverse arc (3,5) decreases to the value of 1.3 minutes. In that case, as shown in Fig. 3, the fastest path between nodes 1 and 5 would be the sequence 1-3-5, with the time invested corresponding to 5.8 (4.5 + 1.3) minutes.

As found in Wen et al. [17], Dijkstra's algorithm, coded as described above, cannot identify the shortest paths for this context of networks with arcs whose lengths may depend on time. Dijkstra's algorithm is also not initially designed to manage other user strategies, such as setting wait times on some nodes to benefit from shorter travel times after waiting.

Among the descriptive data of this new network context is the identification of those arcs (i, j) whose travel time $d_{ij}(t)$ changes according to the exit time of node i. We will assume that the form of variation of $d_{ij}(t)$ corresponds to a step function that changes its value at certain instants (increasing or decreasing) and remains at that level until the next milestone.

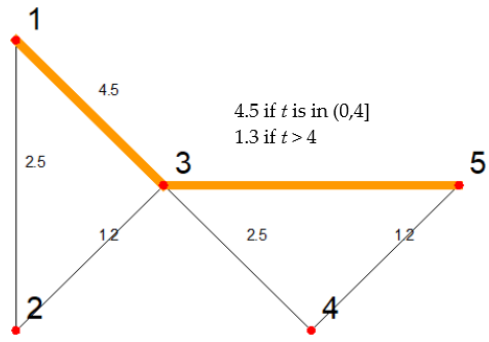

Figure 3: Quickest path between nodes 1 and 5 with time-dependent arcs and no waiting at nodes.

In order to adapt the algorithm of Dijkstra to solve the problem of determining shortest paths in networks with arc lengths that depend on time when starting the trip along the arc, we will make a series of modifications to the original network.

First, we will immerse the planar network in three-dimensional space by raising an axis orthogonal to the base plane for each of the nodes of the network. These axes will indicate the temporal progress at their respective node. For each arc that, starting from node i (at height 0), changes the value of $d_{ij}(t)$ in time t_k, such node i will have to be replicated along its axis, exactly at height t_k. Additionally,

- The distance between nodes i (level 0) and i (level t_k) should be interpreted as waiting time, since geographically there is no change in position.
- The arc (i,j) starting at level 0 indicates that no wait has been carried out at node i by the user before continuing the journey to node j. On the other hand, the arc (i,j) that starts at level t_k indicates that the user has made a wait in node i whose duration is t_k.
- According to the previous assertion, the arc (i,j) that starts at level 0 will be weighted with the value $d_{ij}(0)$, while the arc (i,j) that starts at level t_k will have the value $d_{ij}(t_k)$ as the travel cost.

Furthermore, among the descriptive data of this new network context is the functional characterization of those arcs (i,j) whose travel time $d_{ij}(t)$ changes according to the exit time of node i. We will assume that the form of variation of $d_{ij}(t)$ corresponds to that of a step function that changes its value at certain instants (increasing or decreasing) and staying at that level until the next milestone.

In Fig. 4 has graphically been represented an example of a step function that indicates that the time required to traverse the arc (i,j) is 5, if the start of the trip takes place in time interval [0, 7]; 9, if the start of the trip takes place in time [7, 10]; and so, on until the daily interval [0, 24] is completed.

Let us note that the number of times it would be necessary to replicate a node i along its vertical axis of waiting time will be a function of the number of arcs that start from said node i towards other vertices j and the number of staggered segments $n_i(j)$ that have been considered for explain the behaviour of the function $d_{ij}(t)$. Denoting by *Succ(i)* to the set of vertices j accessible from node i, the number of replications of node i will be, in principle: $\sum_{j \in Succ(i)} n_i(j)$.

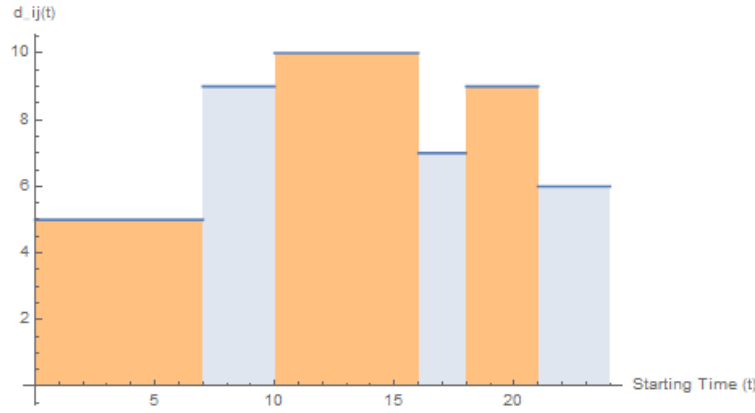

Figure 4: Instance of step function describing $d_{ij}(t)$.

Actually, if we carry out a control of the time progress in each of the vertical axes that start from each node of the network, the number of replicated nodes can be substantially reduced. In fact, it will only make sense to consider the incorporation of the node associated with a travel time $d_{ij}(t_w)$,, obtained after a waiting time t_w at node i, if the sum of both times t_w and $d_{ij}(t_w)$, improves, i.e., is less than the set value without the need to enter waits.

The following example, that will be explained step by step, will explain the proposed methodology. Consider the planar network illustrated in Fig. 5 (8 nodes and 13 arcs), which has previously been embedded in three-dimensional (3D) space. Suppose that origin and destination nodes are labelled 1 and 8, respectively. At the midpoint of each arc of the graph, a numerical value appears, in blue, which corresponds to the time needed to traverse such arc when it is not subject to dependence on the period in which the traversal takes place. In this static context, the fastest path between origin and destination points for these weights is the sequence {1,2,5,7,8} with a total length of 12.5.

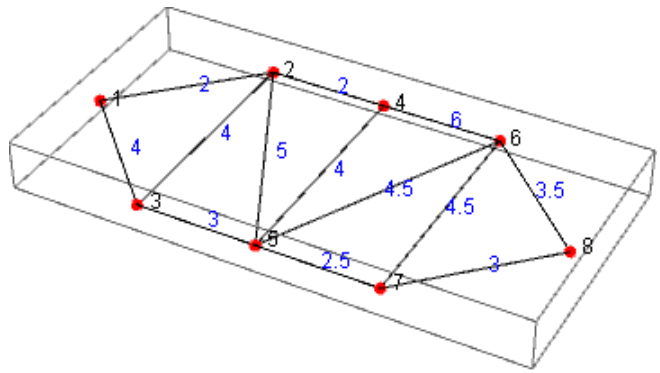

Figure 5: Instance of planar network embedded in 3D-space.

Starting from each node of the network, vertical axes will be established where the new nodes to be replicated can be located. In order to have a reliable control of the time when we

analyse the possible alternatives, we will relocate the intermediate nodes (that is, nodes 2–7) to the heights that correspond on their respective axes with the shortest access time from the origin. In Fig. 6, all 8 nodes have been replicated in three-dimensional space (in red). The origin (1) and destination (8) nodes have not changed their location, while the intermediate nodes have been relocated to the corresponding level with the time necessary to reach them from the origin.

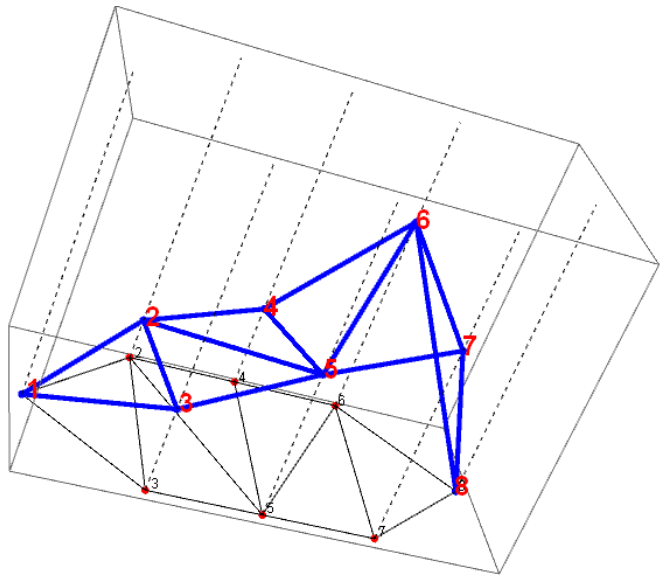

Figure 6: Network with replicated nodes in 3D-space.

Suppose now that two of the three arcs starting from node 4 are time dependent. Specifically, from t = 5, in the arc (4.5) its travel time decreases from 4 to 1.5. For that same mark of t = 5, the arc (4,6) changes its travel time from 6 to 3. Fig. 7 shows the graphs corresponding to the functions $d_{45}(t)$ and $d_{46}(t)$.

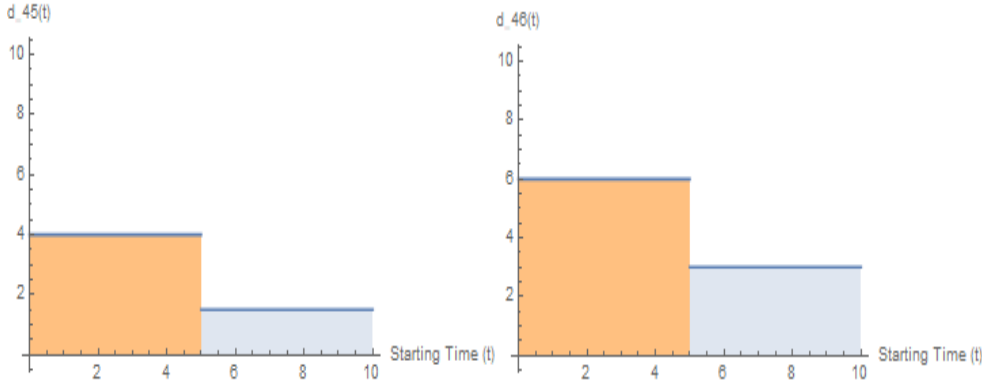

Figure 7: Graphical display of functions $d_{45}(t)$ and $d_{46}(t)$.

Since node 4 is reachable from the origin point (1) in a time t = 4, the user will have two options if node 4 were part of the optimal route to the destination point (8):

1. Immediately explore the arcs starting from node 4 as possible continuation alternatives of an optimal route candidate. This option will not involve modifications in the outgoing arcs from node 4, which graphically is at the elevation level that corresponds to the access time from the origin point.
2. Establish a strategic stop at node 4, with the expectation of obtaining a more competitive travel time on some outgoing arc. This second option forces node 4 to be replicated at the higher vertical level that results after adding the waiting time to the arrival time at the node 4. Node 9, on the same vertical axis of node 4, represents the option to spend one unit of waiting time at node 4 before choosing to continue towards node 5, where the arc (9,5) would be weighted with the value 1.5 instead of 4, or towards node 6, where the arc (9,6) would be weighted with the value 3 instead of 6. Note that it has not been necessary to replicate more times the node 4 because the value changes of the functions $d_{45}(t)$ and $d_{46}(t)$ overlap at the same milestone $t = 4$.

Fig. 8 illustrates the resulting final graph on which, applying a standard Dijkstra's algorithm, the optimal connection route from point 1 to point 8 could be determined, which would be the sequence {1,2,4,9,6,8} with a total length of 11.5. The presence of node 9 in the optimal node sequence indicates that, in this case, the strategy of setting a waiting time at node 4 has been successful.

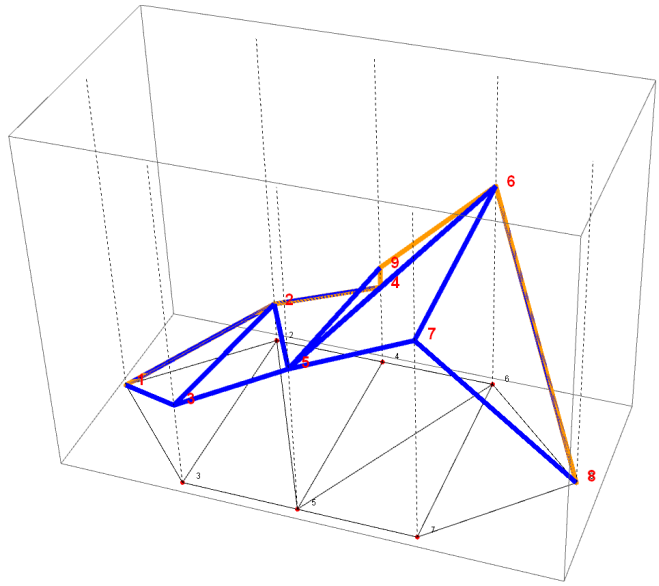

Figure 8: Optimal route connecting points 1 and 8 in 3D-space.

3 AN ADAPTED ALGORITHM

The algorithm presented below includes the necessary modifications to be able to determine the fastest paths between pairs of nodes within a network, with the weights of their arcs being subject to variations according to known or forecast schedules in advance.

Adapted Dijkstra's algorithm with time-dependent arcs

1. Read graph **G= (V, A),** adjacency matrix **Ady** and matrix **D** of displacement costs between pairs of adjacent points.

2. Build vector **f** that will store the minimum displacement costs to the origin from each node of **G** (initially, only the nodes adjacent to the origin will not have infinite value)**.**

3. Build vector **p** that stores the successor nodes to each node of G in the optimal path (initially, only nodes adjacent to the origin can be determined).

4. Initialize set **S** of nodes explored by including only the origin point. **Initialize** set **LIST=G\S.**

5. While LIST is not empty:

5.1 Identify index **j***

$$j^* := Arg\left[\min\{f(j) : j \in LIST\}\right]$$

5.2 If all output arcs from node **j*** do not change in time **then**

5.2.1 Remove index j* from LIST

5.2.2 For each successor **k** of j* included in LIST:

If $f(k) > f(j^*) + d_{j^*k}(0)$ **then**

i. **Update** $f(k) := f(j^*) + d_{j^*k}(0)$

ii. **Update** $p(k) := j^*$

5.3 else For each output arc from **node j*** towards some not visited node that changes over time and for each modification of the weight of this arc**:**

5.3.1 Replicate node j* (let be j*'), updating set V.

5.3.2 Add an arc from j* to j*', weighted with the required wait time at node j*.

5.3.2 Repeat for the new node j*' the same out connections that node j* already had (with the corresponding new weights).

5.3.3 Add an arc from j* to j*', weighted with the required wait time at node j*.

5.3.4 Update set A according to 5.3.2 and 5.3.3.

5.3.5 Include node j*' in set LIST.

6. End

Note that the added programming block mainly corresponds to the section labelled 5.3, and it is activated when time-dependent output arcs are detected. The complexity of this algorithm, that maintains the same structure of the original Dijsktra's algorithm, depends precisely on the number of arcs whose weights are subject to time-dependent changes, as well as the number of changes to be incorporated along the time horizon for each arc.

4 CONCLUSIONS

In this article, the problem of determining the fastest paths with time-dependent arcs in transport networks has been analysed. There are numerous existing bibliographical contributions related to this subject, mainly due to the relevance that this issue acquires in logistics and travel planning for all types of users. Specifically, the methodology presented in Wen et al. [17], where two heuristic methods were proposed to solve the least cost path problem between a pair of nodes with a time-varying road network and a congestion charge. The tool developed by these authors was based on modifications of Dijkstra's algorithm, where a wait ban had been established on the nodes.

The fastest path algorithmic search technique with time-dependent arcs introduced in this contribution follows this methodological line of adaptation of Dijkstra's algorithm to this context, which guarantees a high level of efficiency for the calculation of solutions. On the other hand, the original graph must be conveniently extended, both in number of nodes and in new arcs (some of them unidirectional, which are those that indicate the time progress). It would be possible, however, to limit this growth of the solution container graph, having a reliable control of the time when the possible alternatives are analysed.

ACKNOWLEDGEMENT

This work was in part supported by the Ministerio de Investigación (Spain)/FEDER under grant PID2019-106205GB-I00. This support is gratefully acknowledged.

REFERENCES

[1] Dijkstra, E.W., A note on two problems in connection with graphs. *Numer. Math.*, **1**, pp. 269–271, 1959.
[2] Cormen, T.H., Leiserson, C.E. & Rivest, R.L., *Introduction to Algorithms*, MIT Press and McGraw-Hill: London, 1994.
[3] Toth, P. & Vigo, D., *Vehicle Routing: Problems, Methods, and Applications*, Vol. 18, SIAM, 2014.
[4] Pillac, V., Gendreau, M., Guéret, C. & Medaglia, A.L., A review of dynamic vehicle routing problems. *European Journal of Operational Research*, **225**, pp. 1–11, 2013.
[5] Preuss, T. & Syrbe, J.-H., An integrated traffic information system. *Proc. 6th Int. Conf. Appl. Computer Networking in Architecture, Construction, Design, Civil Eng., and Urban Planning (europIA '97)*, 1997.
[6] Delling, D. & Wagner, D., Time-dependent route planning. *Robust and Online Large-Scale Optimization*, Springer-Verlag, Berlin, pp. 207–230, 2009.
[7] Eglese, R., Maden, W. & Slater, A., A road timetable TM to aid vehicle routing and scheduling. *Comput. Oper. Res.*, **33**, pp. 3508–3519, 2006.
[8] Gendreau, M., Ghiani, G. & Guerriero, E., Time-dependent routing problems: A review. *Computers and Operations Research*, **64**, pp. 189–197, 2015.
[9] Perakis, A.N. & Papadakis, N.A., Minimal time vessel routing in a time-dependent environment. *Transp. Sci.*, **23**, pp. 266–276, 1989.
[10] Norstad, I., Fagerholt, K. & Laporte, G., Tramp ship routing and scheduling with speed optimization. *Transp. Res., Part C*, **19**, pp. 853–865, 2011.

[11] Helvig, C., Robins, G. & Zelikovsky, A., The moving-target traveling salesman problem. *J. Algorithms*, **49**, pp. 153–174, 2003.
[12] Jiang, Q., Sarker, R. & Abbass, H., Tracking moving targets and the non-stationary traveling salesman problem. *Complex Int.*, **11**, pp. 171–179, 2005.
[13] Sutner, K. & Maass, W., Motion planning among time dependent obstacles. *Acta Inf.*, **26**, pp. 93–122, 1988.
[14] Latombe, J.-C., *Robot Motion Planning*, Springer-Verlag: Berlin, 1990.
[15] Fujimura, K., Time-minimum routes in time-dependent networks. *IEEE Trans. Robot Autom.*, **11**, pp. 343–351, 1995.
[16] Dreyfus, S.E., An appraisal of some shortest-path algorithms. *Operations Research*, **17**, pp. 395–412, 1969.
[17] Wen, L., Catay, B. & Eglese, R., Finding a minimum cost path between a pair of nodes in a time-varying road network with a congestion charge. *Eur. J. Oper. Res.*, **236**, pp. 915–923, 2014.