Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 00 (2016) 000–000

Transportation Research Procedia

www.elsevier.com/locate/procedia

19th EURO Working Group on Transportation Meeting, EWGT2016, 5-7 September 2016, Istanbul, Turkey

A multimodal transport network model and efficient algorithms for building advanced traveler information systems

Omar Dib[a,b,*], Marie-Ange Manier[b], Laurent Moalic[b] , Alexandre Caminada[b]

[a]*Technological Research Institute SystemX, 91120 Palaiseau, France*
[b]*OPERA-UTBM, 90010 Belfort, France*

Abstract

Route planning in urban public transport systems constitutes a common decision problem faced by travelers. Therefore, building Advanced Traveler Information Systems (ATIS) that provide passengers with pre-trip information on navigating through the network has become a certain need. Since passengers do not only seek a short-time travel, but they endeavor to optimize other criteria such as cost and effort, an efficient routing system should incorporate a multiobjective analysis for both routes and transport modes. We propose in this paper a new formulation that adequately allows representing a public transit network, as well as, yielding correct results when applying routing algorithms. Based on this formulation, we develop a multicriteria routing algorithm to determine the entire set of nondominated solutions to solve an itinerary planning problem. We introduce also several enhancement strategies to accelerate the algorithm's search process. As transportation modes, we focus on Railway, bus, tram and pedestrian. As optimization criteria, we use travel time, number of transfers and the total walking time. Experimental results have been assessed by solving real life itinerary problems defined on the transport network of the city of Paris and its suburbs. Results indicate that test problems were solved within reasonable amount of time and the new approach is efficient enough to be integrated within a real world journey-planning system.

Peer-review under responsibility of the Scientific Committee of EWGT2016.

Keywords: Multimodal networks; modeling; multiobjective analysis; accelerating strategies

* Omar DIB. Tel.: +33 6 05 51 30 49.
E-mail address: omar.dib@irt-systemx.fr

Peer-review under responsibility of the Scientific Committee of EWGT2016.

1. Introduction

Online services for route planning in transportation networks have become a commodity used daily by millions of commuters. The problem of efficiently computing good journeys presents several modeling and algorithmic challenges, and has been an active area of research in recent years.

In fact, when dealing with transportation systems, we should not consider each transportation mode separately. Rather, we should look to them as one single system with relations and dynamics between its components. For this seek, we need a model reflecting the multimodal nature of the transportation system. This model should also adequately represent each component of the system (stops, routes, trips), as well as, yield correct results when applying routing algorithms.

Routing applications whether they arise in transportation area or other domains such as communication networks, energy and military usually refer for solving Shortest Path Problems (SPP). While solving some routing problems can be done in a straightforward manner, computing shortest paths under certain circumstances is not always an easy task. For instance, solving the one-to-one SPP in static networks can be easily accomplished by applying the well-known algorithm of Dijkstra. On the other side, computing multicriteria shortest paths appears to be more difficult especially in large-scale time-dependent networks

Computing itineraries w.r.t several criteria refers to the multicriteria or Multiobjective Shortest Path Problem (MOSP), a fundamental problem in the field of multiobjective optimization. More precisely, given two journeys j_1 and j_2, we say that j_1 dominates j_2 if there is at least one criterion for which j_1 has a better value than j_2 and there is no criterion for which j_2 has a better value than j_1. A journey j is then called Pareto-optimal if it is not dominated by any other journey. The ultimate goal in multiobjective analysis is therefore to find all Pareto-optimal solutions.

The difficulty in multiobjective optimization stems from the fact that, in many optimization problems, determining the entire set of nondominated solutions is a tedious task since one problem may have a huge number of nondominated solutions (even in case of two objectives). Additionally, and in contrast to single criteria search, one cannot abort the search after finding a first optimal solution. Even after finding all Pareto-optima, search algorithms require a substantial amount of time to guarantee that no further solution exists.

The main goal of this paper is to compute Pareto-paths in multimodal transportation networks. To do so, we propose a new modeling approach allowing efficiently computing multicriteria shortest paths. We then develop a routing algorithm to compute Pareto paths over the proposed modeling approach w.r.t some criteria. We finally introduce several efficient enhancement strategies to improve the performance of the proposed algorithm.

We assess the performance of our work by solving real life itinerary planning problems based on the real data of the French region Il-de-France that includes the city of Paris and its suburbs.

The rest of this paper is structured as follows: in next section, we present some related works. We introduce in Section 3 the new modeling approach and fix some notations. In Section 4 we introduce the multicriteria routing algorithm to solve the MOSP. Experimental results are presented in Section 5. Finally, Section 6 gives some comments and outlines future works.

2. RELATED WORKS

Routing is a widely researched topic in transport systems, mainly because of its relevance to real world applications. The major research effort on this problem relates to two things: modeling transport network and solving routing issues. While the former consists of defining how to represent a transport system, the latter deals with developing algorithms to support routing issues faced by travelers and transport operators.

In terms of modeling, Pyrga et al., (2007) and Delling et al., (2009) have done extensive works to incorporate the multimodality aspect into their models. Liu et al. (2009) proposed a switch point approach to model multimodal transport networks. Van Nes (2002) conducted several researches for designing multimodal transport networks. Ayed et al. (2008) proposed a transfer graph approach for multimodal transport problems. Zhang et al. (2011) introduced a generic method to construct a multimodal transport network representation by using transfer links, which is inspired by the so-called super-network concept. Pyrga et al. (2004) has also done relevant works to generalize a time-expanded model that deals with realistic transfers. Bast et al. (2010) also handled multimodal networks by incorporating predefined transfer arcs between nearby stations.

Although the previously mentioned works are very interesting, they have some limitations. Firstly, they usually disregard important elements presented into the transport system such as platforms where passengers wait, entrance point of stations, transfer inside and outside stations, variable transfer times. Therefore, itineraries resulting from applying routing algorithms would not always imitate real life.

Secondly, such representations usually suffer from significant computational efforts and high memory requirements when loading data and applying routing algorithms Bast et al., (2015).

Thirdly, most models are not flexible enough to handle additional real-world routing problems such as multicriteria shortest paths, dynamic re-routing, handling vehicles' capacities, etc. To overcome such limitations, we introduce in this paper a new generic model for representing multimodal transport networks.

When it comes to routing algorithms, several approaches have been proposed for solving basic and advanced routing problems. For instance, Pajor (2009) adapted the algorithm of Dijkstra to take into account the time dependency and the multimodality aspect of the transport system. Zografos and Konstantinos (2009) described an algorithm for itinerary planning based on dynamic programming. Wang (2008) did a study on handling times and fares in a routing algorithm for public transport. Pyrga et al. (2008) solved the earliest arrival problem on the time-expanded model in a straightforward manner by using a modified version of the algorithm of Dijkstra. Kirchler (2013) introduced a granular tabu search to solve the Dial-A-Ride problem.

Computing multicriteria shortest paths has been also studied recently. For instance, Hamacher et al., (2006) proposed a backward label-setting algorithm for identifying important solutions for the all to one multiple criteria time-dependent shortest path. Modesti et al. (1998) also used a linear utility function that incorporates travel time, ticket cost, and "inconvenience" of transfers. Although the above-mentioned works on handling multicriteria are very significant, however, they have some drawbacks. From one side, they have not been evaluated on real world test instances. From the other side, they usually lie on incomplete model for representing multimodal networks. Therefore, it is crucial to study the issue of multicriteria shortest paths using real world data instances, as well as, on a multimodal formulation that adequately represent the transport system.

3. Modeling Approach

This section considers modeling a multimodal transportation network. It should be clarified that the term multimodal is used in the sense of multiple fixed scheduled transport services. A key difference to static networks is that public transit networks are inherently time-dependent, since certain segments of the network can only be traversed at specific, discrete points in time. As such, the first challenge concerns modeling the timetable appropriately in order to enable the computation of journeys.

Roughly speaking, a timetable consists of a set of stops (such as bus or train platforms), a set of routes (such as bus or train lines), and a set of trips. Trips correspond to individual vehicles that visit the stops along a certain route at a specific time of the day. Trips can be further subdivided into sequences of elementary connections, each given as a pair of (origin/destination) stops and (departure/arrival) times between which the vehicle travels without stopping.

The key modeling in the proposed approach is to model each transportation mode as a separate directed graph. An additional work is then done to integrate all sub-graphs into one larger graph. As a first step of modeling, we introduce three types of nodes that correspond to stations, platforms and departure events.

Although in real life a station may have several access points, we assume in this paper that each station only has one entrance area. A station also comprises a set of platforms where passengers wait for vehicles. An edge is then inserted between a platform and its parent station; its weight represents the minimal time required for accessing that platform from the entrance point of the station.

It is worth to mention that most of representations in the literature disregard platforms. Instead, they only focus on vehicles. However, platforms are essential since transfers inside stations happens between platforms. Moreover, in some routing issues such as evacuations, platforms play an essential role since they give ideas about the capacity of vehicles or even the saturation of the transport system. As a result, we decided with clearly modeling platforms.

Usually, a platform cannot belong to more than one station; however, a station can contain one or several platforms. Each platform has also a type (Bus, railway, tram...) to differentiate between modes. This information can be used by routing algorithms that respect the preferences of the user (i.e. a user prefers only to take the bus along his/her journey).

Since a timetable consists of time-dependent events (e.g., a vehicle departing at a stop) that happen at discrete points in time, we build a space-time graph to unroll time. Roughly speaking, we create a vertex for every event in the timetable that consists of vehicle departing from a platform *x* at *dt* (departure time) and arrives to another platform *y* at *at* (arrival time). Timestamps are inserted into event nodes to account for the departure and arrival times. Event nodes are ordered in the way that a higher-level node refers to an earlier event.

In addition, waiting, boarding and alighting edges are inserted between event nodes and platforms. Additionally, to account for transfers between and inside stations, we inserted transfer edges associated with transfer times between platforms. It is worth to mention that most of the representations in the literature assume that one station has a fixed transfer time. Thereby, transferring between any two platforms inside a station takes the same time. However, such assumption usually yield incorrect results when applying the routing algorithm. Therefore, with the proposed approach, we consider that each station has a variable transfer time.

It is worth to clarify that this model can also be elaborated further to handle additional information such as the capacity of vehicles. For instance, an event node can store an information about the maximum and current capacity of its current vehicle. This information is crucial when dealing with evacuation situations that require rerouting of passengers from one vehicle/mode to another.

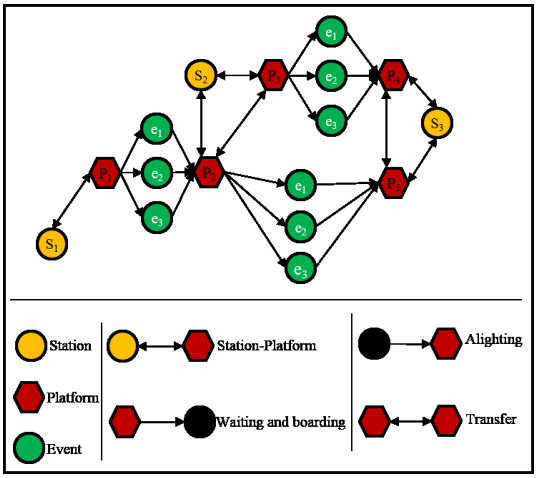

Figure1: Example of modeling 3stations, 5 platforms, 9 events

4. Multicriteria Routing algorithm

After presenting the modeling approach in the previous section, we introduce in this section a routing algorithm for solving the multicriteria shortest path problem. The emerging problem consists of determining the entire set of nondominated paths to go from one station at certain departure time to another station w.r.t the following criteria: i) the total travel time ii) number of transfers iii) the total walking time.

Although scheduled based transportation modes have a time dependency arised from timetable information, the proposed modeling approach allows solving the emerging problem as a variant of the shortest path problem.

As input, the algorithm takes: i) departure station *ds* ii) arrival station *as* and iii) departure time *dt*.

The first step of the algorithm refers to the initialization phase. This phase consists of assigning each platform a list containing one and only one nondominated solution. A nondominated solution is a data structure containing the following information: An integer representing the arrival time denoted *(at)*, an integer representing the number of transfers *(nt)*, an integer related to the walking time *(wkt)*, a reference to a predecessor platform *(fromPlat)* and an integer representing the index of a nondominated solution inside a list (*fromSol*).

Except for platforms belonging to the departure station, labels related to criteria *(at, nt and wkt)* are initialized with big integer values. In addition, the label *fromPlat* is initialized with *null* and the integer *fromSol* is set to *-1*.The above initialization is similar to saying that at the beginning, the algorithm will spend a huge amount of time, transfers and walking time to go from the departure station to all the other stations including the arrival one.

In contrast to most platforms, labels on platforms belonging to the departure station are initialized differently as follows: (*at = dt + (time required to access the platform), nt = 0, wkt = (time required to access the platform), fromPlat = null and fromSol = -1*

This initialization is made differently since computing labels on platforms belonging to the departure station is explicit. As in most shortest path approaches, the proposed algorithm maintains a priority queue, however, elements inside the queue refers in our case to platforms. A platform P_1 is prior than another platform P_2 if labels corresponding to criteria in P_1 have values less than labels corresponding to criteria in P_2. The ultimate goal is to minimize all of the criteria considered. Criteria are compared lexicographically w.r.t the following order: travel time, number of transfers and walking time. Since a platform may contain a set of nondominated solutions, we compare the least value of each criterion among all elements in the set.

At the beginning, the algorithm initializes the priority queue and platforms belonging to the departure station are all added to the queue. After the initializing phase, the algorithm continues to perform some operations until it reaches the stopping criterion.

While in a single criteria shortest path, one can abort the search when a platform belonging the arrival station is reached, we cannot do the same thing in a multiobjective context. A key difference to single objective problem is that in multiobjective context, the algorithm is obliged to visit the entire search space in order to guarantee finding all Pareto optimal paths. Therefore, in this work, the algorithm stops when the priority queue has become empty and not when the algorithm visits a platform belonging to the arrival station.

While the priority queue is not empty, the algorithm takes the first element in the queue (first platform: *fp*) and performs two major operations: i) Handle platforms reached from (*fp*) via event nodes ii) Handle platforms reached from (*fp*) via transfer edges.

For all nondominated solutions existing in the unqueed platform *fp*, the algorithm relaxes all platforms that can be reached from *fp* via events. A platform reached from an event node is denoted by *ap* (arrival platform). The relaxation is performed by creating a candidate nondominated solution that corresponds to the values of the actual nondominated solution at *fp* updated w.r.t the values of the selected event. The algorithm compares the candidate nondominated solution with all the nondominated solutions at the arrival platform *ap* w.r.t each selected event.

During the comparison, a nondominated solution is removed from *ap* if it is dominated by the candidate solution. Otherwise, the candidate solution enters the list of nondominated solutions of *ap*. The candidate nondominated solution is only inserted if it is nondominated by any of the nondominated solutions belonging to *ap*.

By doing so, we ensure, that at each step, the algorithm maintains a feasible list of nondominated solutions to go from the departure station to the reached platform.

It is also worth to mention that when a candidate nondominated solution enters the list of nondominated solutions of a platform, we modify the two labels *fromPlat* and *fromSol*. We set the former label with the unqueud platform *fp* and the latter with the index of the actual nondominated solution at *fp*. These two labels are indispensable when we reconstruct the final dominated paths.

After comparing all nondominated solutions at *fp* with all nondominated at one reached platform *ap*; the platform *ap* enters the priority queue if at least one new nondominated solution has been inserted into its nondominated solution list. Before inserting a platform into the priority queue, we remove it from the queue if it exists to avoid redundancies, as well as, to reorder the element in the queue. In contrast to single objective shortest path, one platform can enter the priority queue several times.

After relaxing platforms reached from events, we do same operations with platforms reached via transfers. Nothing will change here except the values of labels belonging to the candidate nondominated solutions that will be constructed while accomplishing the comparison.

When the priority queue becomes empty, it is guaranteed that the all nondominated paths have been detected between the departure station and all platforms belonging to the arrival stations w.r.t the departure time. Therefore, in order to get the nondominated solutions between the departure station and the arrival station itself we have to only select the nondominated solutions among all nondominated solutions of all platforms belonging to the arrival station.

The last step in the algorithm is to retrieve the paths found. As at each step the algorithm updates labels corresponding to *fromPlat* and *fromSol*, we accomplish a backward search for retrieving paths. We start with the arrival station and for each nondominated solution; we go backward until we reach the departure station.

Input:
- i) directed graph : G(V, E) ii) (departure, arrival) stations : (ds, as) iii) departure time : (dt)
- criteria: C_n ={travel time, number of transfers and walking time}

Output:
- A set of nondominated paths between (ds) and (as) w.r.t (dt) and C_n

Body :
- initialize labels
- initialize priority queue : *queue = PriorityQueue()*
- update labels of platforms belonging to *ds*
- add *ds*'s platforms to the queue
- **while** not queue.empty **do**
  - fp = queue.poll();
  - ndsFp {} = fp.getAllNonDominatedSolutions()
  - Rpe {} = fp.reachablePlatformsViaEvents()
  - Rpt {} = fp.reachablePlatformsViaTransfer()
  - **foreach** ap in Rpe do
    - ndsAp {} = ap.allNonDominatedSolutions()
    - **foreach** nd_1 in ndsFp **do**
      - **foreach** all nd_2 in ndsAp **do**
        - firstEvent = firstEvent (fp, nd_1)
        - canNd = candidatesolution (fp, firstEvent)
        - **if** (canNd dominates nd_2) **then**
          - ndsAp.remove(nd_2)

          else then
          - ndsAp.add(canNd)
        - canNd.setFromPlat(fp)
        - canNd.setFromSol(indexOfNd_1)
        - **end if**
      - **end for**
    - **end for**
    - queue.{remove,add}(ap)
  - **end for**
  - **foreach** ap in Rpt **do**
    - ndsAp {} = ap.getAllNonDominatedSolutions()
    - **foreach** nd_1 in ndsFp **do**
      - **foreach** nd_2 in ndsAp **do**
        - canNd = candidatesolution (fp, transferTime)
        - **if** (canNd dominates nd_2) **then**
          - ndsAp.remove(nd_2)

          else then
          - ndsAp.add(canNd)
        - canNd.setFromPlat(fp)
        - canNd.setFromSol(indexOfNd_1)
        - **end if**
      - **end for**
    - **end for**
    - queue.{remove,add}(ap)
  - **end for**
- **end while**
- **if** path found **then**
  - {retrieve, visualize}paths
- **end if**

Algorithm1: Pseudo-code of routing algorithm

5. Experimental results

To assess the performance of the proposed work, we developed an advanced web-based routing application based on the real data of the French region Île-de-France that includes the city of Paris and its suburbs. Data are provided by the transport organization authority that controls the Paris public transport network and coordinates the different transport companies operating in Île-de-France, mainly the RATP, the SNCF and Optile.

Data comprise geographical information, as well as, timetable information for four transport modes, which are Bus, Metro, Railway, and Tram. More precisely, data encompass 17950 stations; 41047 platforms; 195000 transfers; 303000 trips and 6800000 events for one day.

Original data are provided as text file with respect to the General Transit Feed Specification (GTFS) format. Although we do not discuss implementation issues in this paper, it is worth to mention that efficient and appropriate data structures have been used to guarantee fast access to information when needed and thereby improve the computational effort of this work. The proposed approach has been implemented using Java/Eclipse IDE. The associated runs for solving test problems were performed on an Intel core I5 of 8 GB of RAM.

Since the increased time in computing optimal itineraries decreases the utility of the journey planning services, the computational effort of the proposed approach constitutes a critical success factor for its integration within an online journey planning decision support system.

To assess the computational effort, we applied the proposed algorithm over 10000 routing queries. The start time, departure and arrival stations are uniformly picked at random. In addition, several scenarios have been created to better evaluating the efficiency of the proposed work.

In each scenario, we vary the number of criteria taken into account, as well as, the consideration of one or several enhancement strategies that we will introduce later. Each scenario corresponds to one line in Table1.

For all scenarios, we provide the average and worst case running time. We also compute the number of nondominated solutions resulting from applying the proposed algorithm. Since the running time depends on the computer used, we also provide the number of elements inserted into the priority queue. By doing so, we can analyze more accurately both the impact of the number of criteria considered and the influence of enhancement strategies.

Before analyzing results, it is worth to clarify that several models from the literature suffer from high computational effort when reading and compiling data. However, in our case, the generation and integration of the different sub-networks take less than 12 seconds. This validates the efficiency of the proposed model, as well as, the appropriateness of the data structures used during the implementation phase.

The first line in in table1 indicates that the algorithm optimally computes the whole set on nondominated paths when dealing with two criteria, mainly the travel time and the number of transfers. The average computational time to find Pareto paths is 1.09 seconds. In the worst case, the running time may increase to 3.91 seconds. Results also show that the average number of nondominated solutions in this scenario is 2.21. In another word, the algorithm can provide passengers with several nondominated paths to go from one station to another. In the worst case, the algorithm may result in 5 nondominated paths.

When we add the walking time as a third criterion, results (line #5 in Table1) show that the average running time increases to 20.60 seconds. In the worst case, the algorithm may spend 191.32 seconds to answer users' queries. Results also show that the average number of nondominated solutions in this scenario increases to 5. In the worst case, the algorithm may result in 30 nondominated paths for one routing query.

It can be noticed from these two scenarios that the algorithm's computational effort increases with the number of criteria considered. More we deal with criteria, more the running time increases. This can be explained by the fact the increasing the number of criteria will increase the number of nondominated solutions at each platform. Therefore, the algorithm will spend more time while adding and removing platforms from the priority queue.

It can also be remarqued that the number of elements inserted into the priority increases with the number of criteria. It can also be seen that the size of Pareto front increases with the number of criteria. This can be explained by the fact that more we consider criteria, more the number of nondominated solutions at each platform increases. Consequently, when the priority queue becomes empty, the algorithm will result in more nondominated solutions at the platforms belonging to the arrival station.

We have also noticed that the maximal number of nondominated solutions among all users request is not very big. In another term, there is always a small finite number of nondominated paths between two stations. Theoretically speaking, the number of Pareto optimal solutions in multiobjective context may grow exponentially even when only considering two criteria. However, in this work, which lies in real world transportation data, the number of Pareto optimal solution does not go exponentially with the size of the network.

Analyzing these two scenarios has also indicated that there is no close relation between the number of nondominated solutions that have been found and the running time of the algorithm. In another word, when the algorithm results in many nondominated solutions, there is no guarantee that the running time would also be high. The major factor that determine the algorithm's computational time is the structure of the problem itself.

As can be remarqued, the running time constitutes a bottleneck for the proposed approach to be integrated in real world routing system where users seek fast answers. Therefore, it is crucial to apply some enhancement strategies whereby we enhance the algorithm's performance. Mainly, we have applied four enhancement strategies (S_1, S_2, S_3, S_4). We show in Table1 the impact of each of such strategies over the whole performance of the proposed work.

Table1: Experimental results

Criteria			**Enhancement strategies**				**Running time(ms)**		**# non-dominated solutions**		**# Inserted elements into priority queue**	
α	β	γ	S_1	S_2	S_3	S_4	~	≁	~	≁	~	≁
●	●	-	-	-	-	-	1.09(s)	3.91(s)	2.21	5	113614.63	263569
●	●	-	●	-	-	-	171.01	562.22	1.4	4	61848.25	129147
●	●	-	●	●	-	-	115.56	402.32	1.4	4	42481.62	122195
●	●	-	●	●	●	-	109.5	351.89	1.4	4	41517.68	116892
●	●	-	●	●	●	●	87.96	141.11	1.4	4	29837	86483
●	●	●	-	-	-	-	20.60(s)	191.32(s)	5	30	855552.72	3034166
●	●	●	●	-	-	-	470.01	1334.00	2.86	13	141170.30	336613
●	●	●	●	●	-	-	406.89	1572.32	2.86	13	99738.12	336165
●	●	●	●	●	●	-	309.77	964.70	2.86	13	97851.92	265349
●	●	●	●	●	●	●	246.92	308.80	2.86	13	70253.82	201110

α: total travel time; β: number of transfers; γ: walking time
S_1: Improving dominance relationship; S_2: Exploit earlier results; S_3: removing unnecessary transfer edges; S_4: avoiding unnecessary nodes
~ : average; ≁: worst case

The first enhancement strategy we introduce in this work refers to S_1 ; it consists of enhancing the nondomination relationship. Indeed, while analyzing the behavior of the proposed algorithm, we have realized that in some cases, the domination relationship is not fair. For instance, the algorithm may result in two nondominated itineraries (travel time; number of transfers) as follows: p_1 (25 minutes; 2) p_2 (2 hours; 1). Theoretically, speaking, p_1 and p_2 are nondominated solutions. However, in a realistic situation, a passenger would not prefer a path longer than another path in 1 hour to only avoid one transfer. More precisely, to call two paths p_1 and p_2 nondominated, the difference between travel times should be more than 30 minutes and the difference between the numbers of transfers should be more than 1. Similarly, in a three criteria context, we say that if two nondominated paths p_1 and p_2 have the same number of transfers, p_1 and p_2 are nondominated if the difference between the travel time is more than 30 minutes and the difference between the walking time is more than 5 minutes. Therefore, we say here that it is not fair to spend more than 30 minutes to only gain 5 minute or less in the walking criterion.

The second strategy (S_2) consists of exploiting earlier results. As the algorithm in a multicriteria context may visit one node several times, earlier results found on platforms belonging to the arrival stations may be used as upper bound limits to prune the search space while the algorithm performs its search process. That is, any candidate solution is ignored earlier during the search if it is dominated by any solution belonging to the arrival station.

The third strategy consists of performing a preprocessing phase to remove unnecessary transfer edges inside stations. Indeed, if the time required to go from one platform *p* to another platform *q* via a third platform *v* is less than the time required to go directly from *p* to *q*; then we remove the direct edge *(p,q)* from the graph. By doing so, we ensure thanks to a preprocessing phase that the transfer inside stations is always accomplished according to the shortest path between platforms inside stations. Only transfer edges can be pruned in that way since they are time-independent. Other edges such as those linking platforms together cannot be pruned since their weights is very dependent on the time the platform is visited.

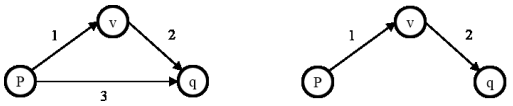

Figure2: removing an unnecessary transfer edge (p,q) since there is a better alternative via the node v

The last strategy refers for adapting a technique from the hyperpath theory to make it works in our multicriteria and time-dependent context. Indeed, we have noticed that, in some cases, the search algorithm may visit unnecessary platforms. For instance if a route is made of several sequent stops (e.g. a bus line), the algorithm does not know if the route may lead to a solution unless it visits all stops along that route. Therefore, we try in this strategy to populate the search algorithm with more knowledge about the routes.

More precisely, we perform a preprocessing technique during the generation phase of the network to build high-level edges between platforms. By doing so, the algorithm becomes able to avoid unnecessary routes earlier in the search process. In Figure3, we show an example of removing four platforms and replacing them by one high-level edge. This edge preserves the time required to go from the platform P_1 and P_6. Moreover, we store in this edge an information about the removed platforms with the time required to go from p1 to each of them.

By doing so, when the algorithm reaches p1, if the destination node is not included in the list of hiding platform in the edge (P_1,P_6) , the algorithm rapidly visits the node P_6. Therefore, the algorithm has succeeded in avoiding relaxing four platforms during its search.

It is worth to mention that only platforms belonging to the same route and that have one and only one adjacent transfer edge are concerned in this strategy. Using this strategy among all platforms requires a hug amount of time in the preprocessing phase and thereby it is not applied in this work.

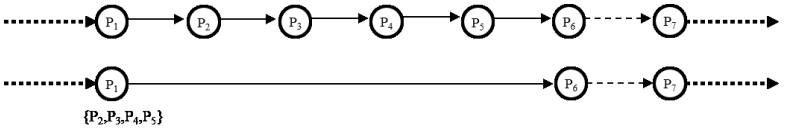

Figure3: removing unnecessary nodes (P_2…P_5) and making them as hiding nodes in the new inserted edge (P_1, P_6)

It is worth to mention that in both strategies (S_3,S_4), which require preprocessing some data in the offline mode to use them while handling users' queries, the time to accomplish the preprocessing phase is not high. While it takes less than to preprocess data in S_3, it takes 4 seconds to perform the preprocessing in S_4. Such reasonable preprocessing time is not prohibitive for using such strategies when the network vastly changes due to some unanticipated factors (e.g. accidents, canceling some trips …). Therefore, using the proposed enhancement strategies is very promising even when the network stochastically changes over the time.

Results in table1 show that implementing the different enhancement strategies has vastly improved the algorithm's performance whether in two or three criteria context. Applying all strategies together has reduced the running time from 1.09 seconds to 87.96 milliseconds when considering two criteria. The running time did also decreased from 20.60 seconds to 246 milliseconds. As can be remarqued, the running time does no longer constitute a bottleneck for the proposed work to be integrated in a real world routing system.

6. Conclusions

We proposed in this paper a new formulation for representing a multimodal transportation network. Based on this formulation, we solved the journey-planning problem that asks for determining the set of nondominated paths to go from one station at certain departure time to arrive to another station. We focused in this paper on the travel time, the number of transfers and the total walking time. As transport modes, we used Railway, Bus Tram and Metro. A multicriteria routing algorithm has been proposed for solving the emerging problem over the proposed modeling approach. Furthermore, several enhancement strategies have been introduced in order to enhance the algorithm's performance. The proposed work has been assessed by solving a wide range of real life itinerary planning problems defined on the Urban Public Transportation System of the French region Ile-De-France that includes the city of Paris and its suburbs. The scope of these tests was to verify that the computational time of the approach in not prohibitive to integrating it within an online journey planning system. In order to make the approach more realistic, we have planned to integrate train delays and other stochastic parameters in future works. Moreover, Transport systems do not only encompass public transportation modes. Other modes such as Bike, and Car Sharing also represent efficient alternatives for many passengers. Integrating such modes into our model is therefore one of our future goals.

Acknowledgments

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

References

Pyrga, Evangelia, et al. "Efficient models for timetable information in public transportation systems." Journal of Experimental Algorithmics (JEA) 12 (2008): 2-4.

Delling, Daniel, Thomas Pajor, and Dorothea Wagner. "Accelerating multi-modal route planning by access-nodes." European Symposium on Algorithms. Springer Berlin Heidelberg, 2009.

Liu, Lu, and Liqiu Meng. "Algorithms of multi-modal route planning based on the concept of switch point." Photogrammetrie-Fernerkundung-Geoinformation 2009.5 (2009): 431-444.

Van Nes, Robertus. Design of multimodal transport networks: A hierarchical approach. TU Delft, Delft University of Technology, 2002.

Ayed, Hedi, et al. "Transfer graph approach for multimodal transport problems." Modelling, Computation and Optimization in Information Systems and Management Sciences. Springer Berlin Heidelberg, 2008. 538-547.

Zhang, Liping, et al. "Traveler Information Tool with Integrated Real-Time Transit Information and Multimodal Trip Planning: Design and Implementation." Transportation Research Record: Journal of the Transportation Research Board 2215 (2011): 1-10.

Pyrga, Evangelia, et al. "Towards realistic modeling of time-table information through the time-dependent approach." Electronic Notes in Theoretical Computer Science 92 (2004): 85-103.

Bast, Hannah, et al. "Fast routing in very large public transportation networks using transfer patterns." European Symposium on Algorithms. Springer Berlin Heidelberg, 2010.

Bast, Hannah, et al. "Route planning in transportation networks." arXiv preprint arXiv:1504.05140 (2015).

Dibbelt, Julian, Thomas Pajor, and Dorothea Wagner. "User-constrained multimodal route planning." Journal of Experimental Algorithmics (JEA) 19 (2015): 3-2.

Zografos, Konstantinos G., Konstantinos N. Androutsopoulos, and Vassilis Spitadakis. "Design and assessment of an online passenger information system for integrated multimodal trip planning." IEEE Transactions on Intelligent Transportation Systems 10.2 (2009): 311-323.

Wang, Jian-Jie. "Optimal paths based on time and fares in transit networks." (2008).

Pyrga, Evangelia, et al. "Efficient models for timetable information in public transportation systems." Journal of Experimental Algorithmics (JEA) 12 (2008): 2-4.

Kirchler, Dominik. Efficient routing on multi-modal transportation networks. Diss. Ecole Polytechnique X, 2013.

Hamacher, Horst W., Stefan Ruzika, and Stevanus A. Tjandra. "Algorithms for time-dependent bicriteria shortest path problems." Discrete optimization 3.3 (2006): 238-254.

Modesti, Paola, and Anna Sciomachen. "A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks." European Journal of Operational Research 111.3 (1998): 495-508.