Public Transportation Algorithm for an Intelligent Routing System

Ander García Gangoiti[1*], Olatz Arbelaitz[2], Oihana Otaegui[1], Pieter Vansteenwegen[3], Maria Teresa Linaza[1]

1. Department of Tourism, Heritage and Creativity, Visual Communication Technologies VICOMTech, Paseo Mikeltegi 57, 2009, San Sebastian, Spain. agarcia@vicomtech.org
2. Department of Computer Architecture and Technology, University of the Basque Country, Spain
3. Centre for Industrial Management, Katholieke Universiteit Leuven, Belgium

KEYWORDS

Multi Path Orienteering Problem Time Windows, Public Transportation, Earliest Arrival Problem.

ABSTRACT

Due to the high complexity of the required calculations, Intelligent Routing Systems have to apply latest Operations Research techniques to be able to create routes efficiently. This paper proposes a solution to the Multi Path Orienteering Problem with Time Windows (MPOPTW), which includes multiple paths to move between locations. The main characteristics of MPOPTW are: the total collected score obtained by visiting locations has to be maximized; not all locations can be visited due to different constraints; and the time required to move from one location to the next one varies according to the departure time, simulating public transportation.

1. INTRODUCTION

The density, size and multimodality of actual urban transportation networks, makes finding an itinerary to move around a city a difficult task for passengers. Intelligent Transport Systems (ITS) provide different solutions to help with this task. These solutions focus generally on solving the problem of finding the best itinerary to move from one place to another according to different criteria (time, money, transportation means …).

We are interested in the tourism domain. When tourists travel to an unknown place, they have to decide [1] which attractions to visit and select the activities to do. Afterwards, these attractions or activities should be time-based filtered and the way to move from one attraction to the next one should be decided.

We are working on a Personalized Electronic Tourist Guide (PET) [2], which is a mobile hand-held device, applying Operations Research (OR) algorithms to create personalized

routes for tourists. Tourists have identified transportation information as one of the most appreciated functionalities of a PET [3]. The algorithm presented in this paper takes into account the available information about the Points of Interest (PoIs) in a city (position, opening time, visit time, tourist "value") and the public transportation network (lines, services, stops, timetable) to calculate a route that maximizes the satisfaction of tourists (total collected score), satisfying several constraints and using public transportation when appropriate.

This paper is organised as follows. Section 2 summarizes the related work. Section 3 describes the algorithm in detail. In Section 4, we present experimental results and we discuss conclusions and further work in Section 5.

2. RELATED WORK

The problem presented in this paper can be divided in two different sub-problems: discovering the earliest arrival time, and the necessary transportation means to a destination location leaving a departure location at a known time; and selecting the most interesting PoIs and the sequence to visit them in order to maximize the collected score and without violating any constraint. The former problem has to be solved several times inside the latter. Thus, efficient algorithms are required in order to obtain a solution in a reasonable time.

The first problem is known as the Earliest Arrival Problem (EAP). Further information about this problem can be found at Pyrga et al. [4]. They have worked on timetable information problems within railway transportation systems. An EAP query (A, B, t_0) consists of a departure station A, an arrival station B, and a departure time t_0. The main objective is to find a connection between A and B leaving not earlier than t_0 and arriving as soon as possible. Pyrga et al. distinguish two versions of this problem: a simplified version, which does not take into account transfer time within services of a station; and a realistic version, with a nonnegative transfer times at least in some stations.

As stated in literature, they have defined a set of trains Z (or buses, ferries…), a set of stations or public transportation stops β, and a set of elementary connections C. Each connection is defined by a tuple $c = (S_1, S_2, t_d, t_a, Z)$, being Z the transportation mean, S_1 the departure stop, S_2 the arrival stop, t_d the departure time from S_1 and t_a the arrival time to S_2. Solving (A, B, t_0) involves finding an itinerary $P = (c_1,c_2,\ldots c_n)$ composed by a sequence of elementary connections c_i and departure and arrival times for each elementary connection.

Moreover, Zografos and Androutsopoulos [5] have implemented an algorithm for itinerary problem in multimodal transportation networks in urban public transportation systems, including a mandatory visit at an intermediate stop. In Hong-Kong, a similar system was tested using a different approach [6].

Regarding the second problem, the root problem is known as Orienteering Problem (OP) [7]. When locations have an associated time window, the problem is called OP with Time Windows (OPTW). The Time Dependent OP (TDO) [8] is an extension of the OP where the time required to travel from a location i to a location j depends on the departure time from location i.

Multi Path Orienteering Problem with Time Windows (MPOPTW) is an extension of OPTW and TDO. In MPOPTW multiple paths are available to move between locations and the time

required to move from one location to the next one can vary according to the departure time. To our understanding, there is no algorithm available to solve the MPOPTW.

PROPOSED SOLUTION

Problem definition

We define L as a set of n locations (each location corresponds to a PoI), each one with an associated score s_i, a visiting duration T_i, opening and closing time $[O_i,C_i]$, arrival time a_i, and leaving time l_i. The traveling time between locations can be defined as an EAP, $t_{ij} = EAP(i,j,l_i)$.

The underlying public multimodal transportation network is an extension of the description available on Pyrgas et al. It is composed by a set of transportation means Z, stops B, and a set of elementary public transportation connections C. Besides, it includes a set of walking connections W as (S_1,S_2,t_{12}) being S_1 the departure node, S_2 the arrival one and t_{12} the time required to go on foot from one to the other. Thus, the complete set of connections of the system is represented as $P = C \cup W$. The set of nodes, N, represents the union of the sets of locations (L) and stops (B) of the system, $N = B \cup L$.

The objective of MPOPTW is to find a route R that maximizes the total collected score not violating the time restriction T_{max} and taking advantage of the available public transportation network. R is composed by an ordered set of locations to visit, $R = (i,j,m,k.....)$.

For each pair of locations (i,j), there is an associated set of connections to move between them, $p^t(i,j) = (p_1,p_2,p_3,.....)$, being t the leaving time from location i. Each connection p_k has a departure node, an arrival node, a departure time, an arrival time and an associated service.

$$p^t(i,j) = \left[\left(i, S_1, l_i, t_{aS_1}, Z_{iS_1}\right), \left(S_1, S_2, t_{dS_1}, t_{aS_2}, Z_{S_1S_2}\right), \ldots, \left(S_n, j, t_{dS_n}, a_j, Z_{S_nj}\right)\right]$$

Algorithm for MPOPTW

The algorithm we propose is divided in three differentiated steps: initialization, Iterated Local Search (ILS)[9] for MPOPTW and Earliest Arrival Problem (EAP).

Initialization

The objective of the initialization is to acquire the required information related to the public transportation network and the different locations of the city. This information is required to initialize the transportation means (Z), the elementary connections (C), the walking connections (W) and the locations. The accuracy of the final results of the algorithm is totally dependant on the quality of the initialization data. Thus, the system requires a continuous update of the data in order to create accurate routes.

Iterated Local Search for MPOPTW

Once the system has been initialized, the second step is executed. The algorithm is based on Iterated local Search (ILS). ILS is a metaheuristic based on a simple idea: iteratively building sequences of solutions generated by an embedded heuristic. This leads to much better solutions than repeating random trials of the same heuristic.

The starting point of an ILS can be an initial solution, s_0, provided by any heuristic able to solve the problem. This heuristic is iteratively executed to find the best possible solution, s^*. To escape from the local optimum, a change or perturbation is applied to s^*, resulting in s'. Then, the local search heuristic is applied to s', in order to obtain a new local optimum, $s^{*'}$. This new solution is compared against the previous best solution, s^*, taking the most suitable of them as the new best solution s^*. The process is then repeated until a termination condition is met. Lourenço et al. [9] introduced ILS.

The implemented local search heuristic is based on an Insert Step that tries to add new visits to a route, one by one [3,10], using two main tools. The first one is *Wait*, the time a tourist has to wait for a location to be opened. The second one is *MaxShift*, which represents the maximum delay in the arrival to a location without causing a route alteration. For each location *i, MaxShift* is calculated as the sum of *Wait* and *MaxShift* of the next location *i+1*, unless its time windows ends before.

For a feasible insertion of a new location *j* between visits *i* and *k*, it is compulsory to satisfy any the constraints. Moreover, the total time consumption in the visit to location *j,* named $shift_j$, needs to be limited to the sum of *Wait* and *MaxShift* of visit *k.* The use of *MaxShift* and *Wait* makes it unnecessary to check the time windows in the remaining locations in a tour, to determine the feasibility of a given local search move. For each location that can be inserted, the shortest insertion time (*shift*) is determined. Moreover, for each of these locations a ratio, which ponders the score of the location with the cost required to visit it, is calculated. The location with the highest ratio is selected for insertion. Then, the Insert Step is repeated until no location can be inserted.

To calculate the cost of going from one location to the next one, the algorithm chooses between public transportation and going on foot, solving the EAP as it will be explained in coming subsections.

The perturbation phase is based on a shake movement that removes consecutive locations from a tour. After the removal, all visits following the removed visits are shifted to the beginning of the tour as much as possible, in order to avoid unnecessary waiting. A tabu list, to avoid removing locations that were recently removed, improves the quality of the results significantly, with low computational cost.

Although it is possible to include advanced acceptance functions based on the search history to decide on the best solution, in our case the new solution obtained by the shake movement is always accepted. The heuristic always continues the search from the perturbed solution, it never returns to the best solution found to continue. This is called ILS with a random walk acceptance criterion [9]. Of course, the best found solution is always kept on memory. Once the termination criteria is met (maximum number of iterations without improvement or maximum allowed time), the system returns the best solution found.

Algorithm for the Earliest Arrival Problem

Each time the algorithm has to decide whether a location is inserted/deleted from a tour, it has to calculate the cost of moving between locations: it has to solve an EAP that includes public transportation and walking connections. .

The walking time only depends on the position of the locations and the pedestrian network of the city. However, other aspects should be taken into account when using public transportation, such as a walking time to and from nearest stops, a waiting time till the transport arrives and a traveling time. Knowing the leaving time from the departure location and the details of the public transportation network (starting time of the services, frequencies, locations of the stops, traveling times between stops), it is possible to calculate the total traveling times and to choose the best option.

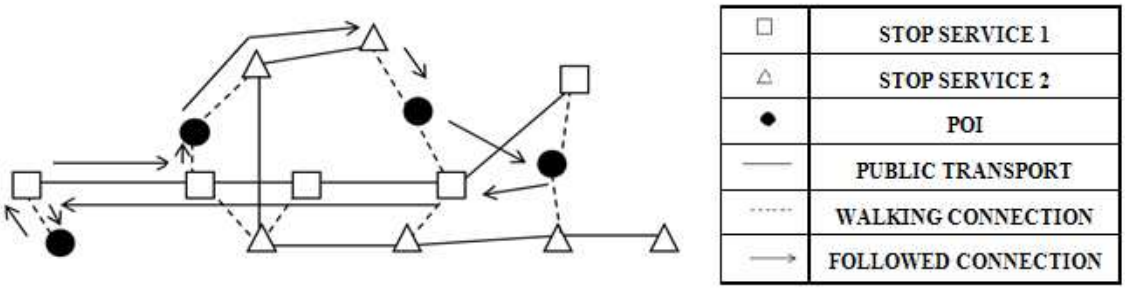

Figure 1- Example of layout of the system

Figure 1 shows a simplified target city where the algorithm could be applied and a possible route leaving and arriving from the PoI located at the bottom left corner. Black dots represent the PoIs spread around the city. Squares and triangles represent the stops of two different public transportation services. Continuous lines represent connections between stops of a service. Discontinuous lines represent walking connections between PoIs and stops shorter than a sensible threshold distance. Although walking connections linking all PoIs together have been excluded for clarity, there is an explicit walking connection between each location.

An algorithm based on Dijkstra's shortest distance has been applied to solve this problem. A time-dependent approach with simple transfer times has been applied due to its better performance. Each node (location/stop) of the system has a group of labels with its arrival time, its penalized arrival time (both of them initialized to infinite) and the path followed to reach it. The algorithm has two sets of nodes: the set of settled nodes, S (nodes whose shortest distances from the source has been found), and the set of unsettled nodes, Q.

The algorithm has four different steps, summarized in Figure 2, which are executed until the shortest distance to all nodes is found. During the initialization step, the distance to all nodes is set to infinite, Q is initialized having all the nodes and S as an empty set. The starting location is set as the initial node, setting its shortest distance to 0, extracting it from Q and adding it to S.

Then, starting from the initial node, its neighboring nodes are obtained. On the third step, knowing the departure time from the initial node, the earliest arrival time is calculated for each of the neighbors. Moreover, if the transportation mean changes, a time penalization is added to the earliest arrival time. If this penalized time improves the previously existing shortest distance, the shortest distance and the path to arrive to the node are updated.

$d = (\infty)$

```
u = i
Q= (N-u)
S = (u)
while Q is not empty
|      find shortest distance to neighbors (u)
|      u = extractMinimum (Q)
|      S = S + u
|      Q = Q - u
end
```

Figure 2- Diagram of the Dijkstra's shortest path based EAP algorithm

A penalization time is added in case of any of the following changes: walking connection to public transportation connection; public transportation connection to walking connection; and line change from one public transportation connection to another public transportation connection. The inclusion of a time penalization of three minutes avoids choosing routes which make users to change transportation for a time saving of only some minutes.

Equations below show the process followed to solve EAP for two connecting nodes. The shortest path is the minimum among the time required to go on foot (t_w) or by public transportation (t_{pt}). A time penalization *(tp)* is introduced as mentioned before. For public transportation connection, the link with a departure equal or later than the departure time from the first node, and the smallest penalized arrival time is chosen.

$$EAP\ (A, B, l_a) = \min(t_w, t_{pt})$$

$$t_w = l_a + t_{ab} + tp$$

$$t_{pt} = c_i(A, B, t_d, t_{a,} Z_i) \Rightarrow \nexists\, c'_i(A, B, t'_d, t'_a, Z'_i) \begin{cases} t'_d > l_a \\ t'_a + tp' < t_a + tp \\ tp, tp' = \begin{cases} 0 & if \neq transport\ change \\ penalization & otherwise \end{cases} \end{cases}$$

If the shortest distance is updated and the connection involves a non previously taken public transportation service, the system obtains the arrival time to the remaining stops of the service in order to shorten the total execution time. The arrival time is checked and the shortest distance is updated in case of an improvement.

Finally, the node with the smallest shortest distance is extracted from *Q*, added to *S*, and taken as the initial node. Then, the first step is executed again. The procedure is repeated until *Q* is empty, meaning the shortest distance to all nodes has been found.

REAL PROBLEM AND RESULTS

The algorithm has been tested in San Sebastian, which is a medium size city (200.000 habitants). The company in charge of the public transportation network has provided real data about stops, lines, frequencies and traveling times. This information has been stored in a local database with the information about several PoIs of the city. Basic public transportation connections and walking distance between PoIs and stops have been automatically calculated. The structure of the database is shown in Figure 3. Stops, PoIs and distances between PoIs, stops and PoIs-stops have been differentiated in order to ease the management of this information.

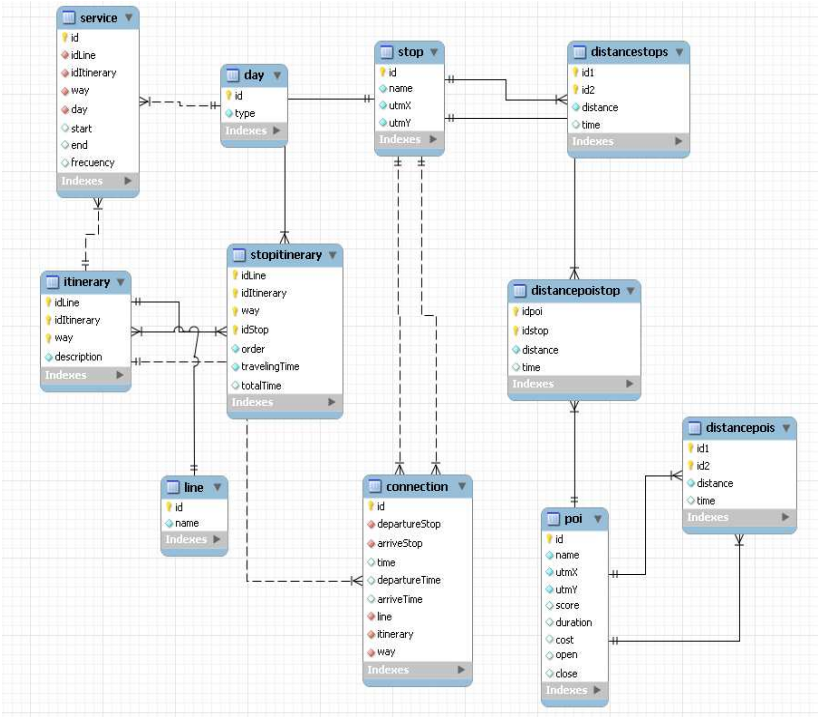

Figure 3 - Diagram of the database

The system includes 467 public transportation stops, 26 lines, 66.650 public transportation connections between stops and 1.642 walking connections between stops (shorter than 200 meters). The algorithm has been coded in Java 1.6. All the tests have been run on a PC with Microsoft Windows XP, Java 1.6, 2 GB and a Intel Core 2 Quad Q6600 (2.4 Ghz) CPU.

Including an algorithm able to solve the EAP inside an algorithm to solve MPOPTW, creates a high calculation demand. For example, initial testing on problems with 5 PoIs, showed that exploring the solution space with the ILS algorithm implies that the EAP is solved around 3000 times, with a final calculation time longer than 10 minutes.

Thus, some optimization techniques have been implemented. Reordering the order of the instructions of the ILS algorithm reduced the number of times EAP need to be solved nearly ten times. Applying cache techniques storing the EAP calculations already made, reduced the execution time nearly in the same proportion. The final implementation of the algorithm is able to solve instances with up to 20 PoIs and the above mentioned public transportation network in less than one minute.

CONCLUSIONS AND FURTHER WORK

The application of advanced algorithms and methaheuristics from Operations Research and timetable information systems, allows developing advanced intelligent routing systems. Within a tourist domain with a finite number of locations and their related opening times, visiting duration and associated score, the system presented in this paper is able to create personalized routes maximizing the total collected score without violating a time restriction set by the user and using the public transportation network.

In order to improve the efficiency of the algorithm, different research approaches will be taking into consideration, such as parallelization techniques to make use of the multi-core capabilities of the actual processors; inserting an initial shortest distance calculating step for possible departure times, in order to avoid its calculation in real time; and the application of latest speed-up techniques from timetable information systems.

ACKNOLWGEMENT

Authors would like to thank the Basque Government for partially funding this work through the neurebide and etourgune projects and to the Centre for Industrial Management of the Katholieke Universiteit Leuven for hosting Ander Garcia as a guest researcher during 2008. Pieter Vansteenwegen is a post-doctoral research fellow of the "Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO)".

REFERENCES

[1] Brown, B., Chalmers, M. (2003). Tourism and mobile technology, in *8th European Conference on Computer Supported Cooperative Work,* Helsinki, Finland*, pp.* 335-354.

[2] Garcia, A., Linaza, M., Arbelaitz, O., Vansteenwegen, P. (2009). Intelligent routing system for a personalised electronic tourist guide, in *Information and Communication Technologies in Tourism 2009*, Springer, Amsterdam, The Netherlands, pp. 185-197.

[3] Vansteenwegen, P. (2008) Planning in tourism and public transportation. PhD dissertation, Katholieke Universiteit Leuven, Centre for Industrial Management, Belgium, ISBN: 978-90-5682-949-0.

[4] Pyrga, E., Schulz, F., Wagner, D., Zaroliagis, C. (2008). Efficient models for timetable information in public transportation systems, *Journal on Experimental Algorithmics*, , vol.12, pp 1-39.

[5] Zografos, K. G., & Androutsopoulos, K. N. (2008). Algorithms for Determining Optimum Itineraries in a Multimodal Urban Transportation Network, *IEEE Transactions on Intelligent Transportation Systems*, Tokyo, Japan, vol. 9, pp 175-184.

[6] Chiu, D. K. W., Lee, O., & Leung, H. F. (2005). A Multi-Modal Agent Based Mobile Route Advisory System for Public Transport Network, in *38th Annual Hawaii International Conference on System Sciences (HICSS'05),* Hawaii, *vol.3 , pp. 92.2.*

[7] Tsiligirides, T. (1984). Heuristic methods applied to orienteering, *The Journal of the Operational Research Society, vol. 35*(9), pp. 797-809.

[8] Fomin, F. V., & Lingas, A. (2002). Approximation algorithms for time-dependent orienteering, *Information Processing Letters*. Elsevier, vol. 83, pp. 57-62.

[9] Lourenco, H.R., Martin O., and Stuetzle T. (2002). Iterated Local Search. In: F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics., Kluwer Academic Publishers, Norwell, MA, pp. 321-353.

[10] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D. (2009). Metaheuristics for tourist trip planning. In: Geiger, M.,Habenicht, W., Sevaux, M., Sorensen, K. (Eds.), Metaheuristics in the Service Industry. Lecture Notes in Economics and Mathematical Systems. Springer Verlag, pp. 15-31.